Understanding and Incorporating Mathematical Inductive Biases in Neural Networks

by

Marc Finzi

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May, 2023

Professor Andrew Gordon Wilson

Acknowledgments

I would like to express my deepest gratitude to my advisor, Andrew Wilson, for accepting me as a student and continuously believing in my potential. I am grateful for the numerous thoughtful discussions and valuable feedback on my work, as well as his guidance to help me become a better writer. His dedication to cultivating a supportive and nurturing lab environment has been central to my growth as a researcher, and I am grateful for his effort towards launching my career post PhD.

A heartfelt thank you to Pavel and Polina for providing moral support during the early stages of my PhD journey, and making the experience altogether much more enjoyable. Thank you also to Wesley and Sam for helping to create the positive lab culture that we have, and for being great collaborators. I am grateful to Nate for our intellectually stimulating conversations and the enjoyable debates about various topics in machine learning. These discussions have enriched my understanding of the field and have made my time in the lab both productive and fun.

A thank you to my coauthors Alex, Greg, Sanyam, Sanae, and Micah for their dedication and hard work brought to our joint projects. A special thank you to Andres enough for his unwavering persistence and for being a wonderful project partner. His camaraderie has made the otherwise solitary research process much more enjoyable, as well as his effort in turning me into a gym bro outside of the lab.

Finally, I want to express gratitude to my parents for their unwavering encouragement and support throughout my PhD journey, always helping me to figure out how to best achieve whatever my goals were. Their love and belief in me have provided the motivation and strength necessary to navigate the challenges and triumphs of this process.

Thank you all (and to those I have not mentioned) for being a part of my PhD journey and for contributing to the completion of this thesis. I am truly grateful for your guidance, friendship, and encouragement.

Abstract

To overcome the enormous sample complexity of deep learning models, we can leverage basic elements of human and scientific knowledge and imbue these elements into our models. By doing so, we can short-circuit the thousands of years of evolutionary development that has enabled such rapid learning in humans, and the development of science which provides a framework to fit new knowledge into. In this work I develop new methods for incorporating mathematical inductive biases into our models, biasing them towards solutions that reflect our priors and our knowledge. This work helps to broaden the scope and automation of equivariant model construction across diverse domains, uncover the role of inductive biases in learning and generalization, and developing new machine learning models for scientific applications, capturing relevant scientific knowledge.

Contents

List of Figures

List of Tables

Appendices

1 | Introduction

The ability to learn from data and generalize to new, previously unseen situations is a critical aspect of artificial intelligence. Deep learning models have demonstrated impressive performance in various domains, ranging from computer vision to natural language processing. However, their success often comes at the cost of requiring vast amounts of data and computational resources. This limitation raises a fundamental question: can we imbue our models with basic elements of human and scientific knowledge to accelerate learning and improve generalization? The answer to this question is the central focus of this thesis, as we explore new methods for incorporating mathematical inductive biases into our models, biasing them towards solutions that reflect our priors and our knowledge.

The rich array of symmetries and equivariances found in nature has long been a source of inspiration for researchers seeking to build more robust and efficient models. By encoding these properties directly into our models, we can potentially achieve faster learning, better generalization, and enhanced interpretability. In this thesis, we start by developing a completely general algorithm for constructing equivariant layers of matrix groups called Equivariant-MLP, reducing the task of designing an equivariant network (at least for low dimensional data) into a purely computational problem. With Equivariant-MLP we help lower the barrier to constructing efficient learners in diverse domains such as dynamical systems, particle physics, and robotics.

Striking the right balance between flexibility and inductive biases remains an ongoing challenge, and like other structures, symmetries are often broken by the messy realities of real data. To address this tension, we introduce Residual Pathway Priors (RPPs), a method to convert hard architectural constraints into soft probabilistic priors. RPP guides models towards structured solutions while retaining the ability to capture additional complexity, and helps encode our beliefs in the Bayesian sense. We demonstrate the broad applicability of RPPs across various domains, including reinforcement learning, where our approach outperforms baseline model-free agents and improves learned transition models for model-based RL.

To better understand the impact of equivariance across different architectures and training methods, we introduce the Lie derivative. The lie derivative is a powerful mathematical tool for measuring equivariance and enables us to identify and isolate failures of equivariance as they are produced by particular layers in a network. By analyzing hundreds of pretrained models, we reveal surprising insights into the relationship between model size, accuracy, and equivariance. Our findings challenge traditional narratives and highlights the impact of better training strategies on the extent to which equivariance is respected.

Finally, to gain a deeper understanding of why deep learning works, we develop a new method for proving PAC-Bayes generalization bounds on deep neural networks. This approach yields state-of-the-art generalization bounds on image tasks, and allows us to better understand the role of model size, transfer learning, equivariance, and the extent to which structures in the model align with structures in the data. Our findings reveal that large models can be compressed to a much greater extent than previously known, supporting Occam's razor and providing new insights into the interplay between inductive biases and generalization.

In conclusion, this thesis represents a significant step forward in our understanding of how to incorporate mathematical inductive biases into deep learning models. By broadening the scope and automation of equivariant model construction, uncovering the role of inductive biases in learning and generalization, and developing new machine learning models for scientific applications that capture relevant scientific knowledge, we hope to not only push the boundaries of what is possible with deep learning but also inspire new avenues of research in the quest for artificial intelligence that learns rapidly, generalizes effectively, and respects the principles of the natural world.

2 General and Automated Equivariant Model Construction with Equivariant-MLP

Symmetries and equivariance are fundamental to the generalization of neural networks on domains such as images, graphs, and point clouds. Existing work has primarily focused on a small number of groups, such as the translation, rotation, and permutation groups. In this chapter we provide a completely general algorithm for solving for the equivariant layers of matrix groups. In addition to recovering solutions from other works as special cases, we construct multilayer perceptrons equivariant to multiple groups that have never been tackled before, including $\mathrm{O}(1,3)$, $\mathrm{O}(5)$, $\mathrm{Sp}(n)$, and the Rubik's cube group. Our approach outperforms non-equivariant baselines, with applications including particle physics and dynamical systems. We release our software library to enable researchers to construct equivariant layers for arbitrary matrix groups.

This chapter is adapted from the paper "A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups", which originally appeared at ICML 2021 and is joint work with Max Welling, and Andrew Gordon Wilson.

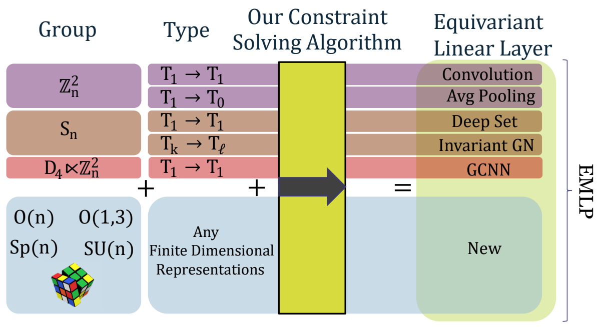


Figure 2.1: We provide a general and efficient method for solving equivariance constaints. For particular symmetry groups and type signatures, we recover other well known equivariant layers while also enabling application to new groups and representations.

2.1 Introduction

As machine learning has expanded to cover more areas, the kinds of structures and data types we must accommodate grows ever larger. While translation equivariance may have been sufficient for working with narrowly defined sequences and images, with the expanding scope to sets, graphs, point clouds, meshes, hierarchies, tables, proteins, RF signals, games, PDEs, dynamical systems, and particle jets, we require new techniques to exploit the structure and symmetries in the data.

In this work we propose a general formulation for equivariant multilayer perceptrons (EMLP). Given a set of inputs and outputs which transform according to finite dimensional representations of a symmetry group, we characterize all linear layers that map from one space to the other, and provide a polynomial time algorithm for computing them. We release our library, along with documentation, and examples.

Figure 2.1 illustrates how the convolutional layers of a CNN [LeCun et al. 1989], the permutation equivariant deep sets [Zaheer et al. 2017], graph layers [Maron et al. 2018], and layers for networks equivariant to point clouds [Thomas et al. 2018], all arise as special cases of our more general algorithm.

We summarize our contributions as follows:

- We prove that the conditions for equivariance to matrix groups with arbitrary linear representations can be reduced to a set of $M + D$ constraints, where M is the number of discrete generators and D is the dimension of the group.

- We provide a polynomial time algorithm for solving these constraints for finite dimensional representations, and we show that the approach can be accelerated by exploiting structure and recasting it as an optimization problem.
- With the addition of a bilinear layer, we develop the Equivariant MultiLayer Perceptron (EMLP), a general equivariant architecture that can be applied to a new group by specifying the group generators.
- Demonstrating the generality of our approach, we apply our network to multiple groups that were previously infeasible, such as the orthogonal group in five dimensions $\mathrm{O}(5)$, the full Lorentz group $\mathrm{O}(1,3)$, the symplectic group $\mathrm{Sp}(n)$, the Rubik's cube group, *with the same underlying architecture*, outperforming non equivariant baselines.

2.2 Related Work

While translation equivariance in convolutional neural networks [LeCun et al. 1989] has been around for many years, more general group equivariant neural networks were introduced in Cohen and Welling [2016a] for discrete groups with GCNNs. There have been a number of important works generalizing the approach to make use of the irreducible group representations for the continuous rotation groups $\mathrm{SO}(2)$ [Cohen and Welling 2016b; Esteves et al. 2017; Marcos et al. 2017], $\mathrm{O}(2)$ [Weiler and Cesa 2019a], $\mathrm{SO}(3)$ [Thomas et al. 2018; Weiler et al. 2018; Anderson et al. 2019], $\mathrm{O}(3)$ [Smidt et al. 2020] and their discrete subgroups.The requirements and complexity of working with irreducible representations has limited the scope of these methods, with only one example outside of these two rotation groups with the identity component of the Lorentz group $\mathrm{SO}^+(1,3)$ in Bogatskiy et al. [2020].

Others have used alternate approaches for equivariance through group FFTs [Cohen et al. 2018b], and regular group convolution [Worrall and Welling 2019; Bekkers 2019; Finzi et al. 2020]. These methods enable greater flexibility; however, achieving equivariance for continuous groups with the regular representation is fundamentally challenging, since the regular representation is infinite dimensional.

Meanwhile, the theoretical understanding and practical methods for equivariance to the permutation group S_n have advanced considerably for the application to sets [Zaheer et al. 2017], graphs [Maron et al. 2018], and related objects [Serviansky et al. 2020]. Particular instances of equivariant networks have been shown to be *universal*: with a sufficient size these networks can approximate equivariant functions for the given group with arbitrary accuracy [Maron et al. 2019; Ravanbakhsh 2020; Dym and Maron 2020].

Despite these developments, there is still no algorithm for constructing equivariant networks that is completely general to the choice of symmetry group or representation. Furthest in this direction are the works of Lang and Weiler [2020], Ravanbakhsh et al. [2017], and van der Pol et al. [2020b] with some of these ideas also appearing in Wood and Shawe-Taylor [1996]. Based on the Wigner-Eckert theorem, Lang and Weiler [2020] show a general process by which equivariant convolution kernels can be derived for arbitrary compact groups. However this process still requires considerable mathematical legwork to carry out for a given group, and is not applicable beyond compact groups. Ravanbakhsh et al. [2017] show how equivariance can be achieved by sharing weights over the orbits of the group, but is limited to regular representations of finite groups. Unlike Lang and Weiler [2020] and Ravanbakhsh et al. [2017], van der Pol et al. [2020b] present an explicit algorithm for computing equivariant layers. However, the complexity of this approach scales with the size of the group and quickly becomes too costly for large groups and impossible for continuous groups like $\mathrm{SO}(n), \mathrm{O}(1,3)$, $\mathrm{Sp}(n)$, and $\mathrm{SU}(n)$.

2.3 Background

In order to present our main results, we first review some necessary background on group theory. Most importantly, symmetry groups can be broken down in terms of discrete and continuous generators, and these can act on objects through group and Lie algebra representations.

Finite Groups and Discrete Generators. A group G is finitely generated if we can write each element $g \in G$ as a sequence from a discrete set of generators $\{h_1, h_2, ...h_M\}$ and their inverses $h_{-k} = h_k^{-1}$. For example we may have an element $g = h_1 h_2 h_2 h_1^{-1} h_3$ and can be written more compactly $g = \Pi_{i=1}^{N} h_{k_i}$ for the integer sequence $k = [1, 2, 2, -1, 3]$.

All finite groups, like the cyclic group $\mathbb{Z}_n$, the dihedral group D_n, the permutation group S_n, and the Rubik's cube group can be produced by a finite set of generators. Even for large groups, the number of generators is *much* smaller than the size of the group: 1 for $\mathbb{Z}_n$ of size n, 2 for S_n of size $n!$, and 6 for the cube group of size 4×10^{19}.

Continuous Groups and Infinitesimal Generators. Similarly, Lie theory provides a way of analyzing continuous groups in terms of their *infinitesimal* generators. The Lie Algebra $\mathfrak{g}$ of a Lie Group G (a continuous group that forms a smooth manifold) is the tangent space at the identity $\mathfrak{g} := T_{\mathrm{id}}G \subseteq \mathbb{R}^{n \times n}$, which is a vector space of infinitesimal generators of group transformations from G. The exponential map $\exp : \mathfrak{g} \rightarrow G$ maps back to the Lie Group and can be understood through the series: $\exp(A) = \sum_{k=0}^{\infty} A^k / k!$

A classic example is the rotation group $G = \mathrm{SO}(n)$ with matrices $\mathbb{R}^{n \times n}$ satisfying $R^\top R = I$ and $\det(R) = 1$. Parametrizing a curve $R(t)$ with $R(0) = I$, $R'(0) = A$, one can find the tangent space by differentiating the constraint at the identity. The Lie Algebra consists of antisymmetric matrices: $\mathfrak{so}(n) = T_{\mathrm{id}}\mathrm{SO}(n) = \{A \in \mathbb{R}^{n \times n} : A^\top = -A\}$.

Given that $\mathfrak{g}$ is a finite dimensional vector space ($D = \dim(\mathfrak{g}) = \dim(G)$), its elements can be expanded in a basis $\{A_1, A_2, ..., A_D\}$. For some Lie Groups like $\mathrm{SO}(n)$, the orientation preserving isometries $\mathrm{SE}(n)$, the special unitary group $\mathrm{SU}(n)$, the symplectic group $\mathrm{Sp}(n)$, the exponential map is surjective meaning all elements $g \in G$ can be written in terms of this exponential $g = \exp(\sum_i \alpha_i A_i)$ with a set of real valued coefficients $\{\alpha_i\}_{i=1}^D$. But in general for other Matrix Groups like $\mathrm{O}(n)$, $\mathrm{E}(n)$, and $\mathrm{O}(1,3)$, exp is not surjective and one can instead write $g = \exp(\sum_i \alpha_i A_i)\Pi_{i=1}^N h_{k_i}$ as a product of the exponential map that traverses the identity component and an additional collection of discrete generators (see subsection A.1.8).

Group Representations. In the machine learning context, a group element is most relevant in how it acts as a transformation on an input. A (linear finite dimensional) group *representation* $\rho : G \to \mathrm{GL}(m)$ associates each $g \in G$ to an invertible matrix $\rho(g) \in \mathbb{R}^{m\times m}$ that acts on $\mathbb{R}^m$. The representation satisfies $\forall g_1, g_2 \in G : \rho(g_1 g_2) = \rho(g_1)\rho(g_2)$, and therefore also $\rho(g^{-1}) = \rho(g)^{-1}$. The representation specifies how objects transform under the group, and can be considered a specification of the *type* of an object.

Lie Algebra Representations. Mirroring the group representations, Lie Groups have an associated representation of their Lie algebra, prescribing how infinitesimal transformations act on an input. A Lie algebra representation $d\rho : \mathfrak{g} \to \mathfrak{gl}(N)$ is a **linear** map from the Lie algebra to $m \times m$ matrices. An important result in Lie Theory relates the representation of a Lie Group to the representation of its Lie Algebra

$$\forall A \in \mathfrak{g} : \quad \rho(e^A) = e^{d\rho(A)} \tag{2.1}$$

Tensor Representations. Given some base group representation ρ, Lie Algebra representation $d\rho$, acting on a vector space V, representations of increasing size and complexity can be built up through the tensor operations dual $(*)$, direct sum $(\oplus)$, and tensor product $(\otimes)$, as detailed in Table 2.1.

Acting on matrices, $\oplus$ is the direct sum which concatenates the matrices on the diagonal $X \oplus Y = \begin{bmatrix} X & 0 \\ 0 & Y \end{bmatrix}$, and facilitates multiple representations which are acted upon separately. The $\otimes$ on matrices is the Kronecker product, and $\bar{\oplus}$ is the *Kronecker sum*: $X \bar{\oplus} Y = X \otimes I + I \otimes Y$. V^* is

OP	ρ	$d\rho$	V
$*$	$\rho(g^{-1})^\top$	$-d\rho(A)^\top$	V^*
$\oplus$	$\rho_1(g) \oplus \rho_2(g)$	$d\rho_1(A) \oplus d\rho_2(A)$	$V_1 \oplus V_2$
$\otimes$	$\rho_1(g) \otimes \rho_2(g)$	$d\rho_1(A) \bar{\oplus} d\rho_2(A)$	$V_1 \otimes V_2$

Table 2.1: Different ways of combining group representations, shown for dual, direct sum, tensor product and the corresponding vector space, group representation and Lie algebra representation.

the dual space of V. The tensor product and dual are useful in describing linear maps from one vector space to another. Linear maps from $V_1 \to V_2$ form the vector space $V_2 \otimes V_1^*$ and have the corresponding representation $\rho_2 \otimes \rho_1^*$.

We will work with the corresponding vector spaces and representations interchangeably with the understanding that the other is defined through these composition rules. We abbreviate many copies of the same vector space $\underbrace{V \oplus V \oplus \ldots \oplus V}_{m}$ as mV. Similarly we will refer to the vector space formed from many tensor products $T_{(p,q)} = V^{\otimes p} \otimes (V^*)^{\otimes q}$ where $(\cdot)^{\otimes p}$ is the tensor product iterated p times. Following the table, these tensors have the group representation $\rho_{(p,q)}(g) = \rho(g)^{\otimes p} \otimes \rho^*(g)^{\otimes q}$, and the Lie algebra representation $d\rho_{(p,q)}(A) = d\rho(A)^{\bar{\oplus} p} \bar{\oplus} d\rho^*(A)^{\bar{\oplus} q}$. We will abbreviate T_{p+q} for $T_{(p,q)}$ when using orthogonal representations ($\rho = \rho^*$), as the distinction between V and V^* becomes unnecessary.

2.4 Equivariant Linear Maps

In building equivariant models, we need that the layers of the network are equivariant to the action of the group. Below we characterize all equivariant linear layers $W \in \mathbb{R}^{N_2 \times N_1}$ that map from one vector space V_1 with representation ρ_1 to another vector space V_2 with representation ρ_2 for a matrix group G. We prove that the infinite set of constraints can be reduced to a finite collection without loss of generality, and then provide a polynomial-time algorithm for solving the constraints.

2.4.1 The Equivariance Constraint

Equivariance requires that transforming the input is the same as transforming the output:

$$\forall x \in V_1, \forall g \in G : \qquad \rho_2(g)Wx = W\rho_1(g)x.$$

Since true for all x, $\rho_2(g)W\rho_1(g)^{-1} = W$, or more abstractly:

$$\forall g \in G : \quad \rho_2(g) \otimes \rho_1(g^{-1})^\top \mathrm{vec}(W) = \mathrm{vec}(W) \tag{2.2}$$

where vec flattens the matrix into a vector. $\rho_1(g^{-1})^\top$ is the dual representation $\rho_1^*(g)$, and so the whole object $\rho_2(g) \otimes \rho_1(g^{-1})^\top = (\rho_2 \otimes \rho_1^*)(g) = \rho_{21}(g)$ is a representation of how g acts on matrices mapping from $V_1 \to V_2$.

While equation (2.2) is linear, the constraint must be upheld for each of the possibly combinatorially large or infinite number of group elements in the case of continuous groups. However, in the following section we show that these constraints can be reduced to a finite and small number.

2.4.2 General Solution for Symmetric Objects

Equation (2.2) above with $\rho = (\rho_2 \otimes \rho_1^*)$ is a special case of a more general equation expressing the symmetry of an object v,

$$\forall g \in G : \quad \rho(g)v = v \tag{2.3}$$

Writing the elements of G in terms of their generators: $g = \exp(\sum_i^D \alpha_i A_i)\Pi_{i=1}^N h_{k_i}$. For group elements with $k = \varnothing$, we have

$$\forall \alpha_i : \quad \rho\big(\exp(\sum_i \alpha_i A_i)\big)v = v$$

Using the Lie Algebra - Lie Group representation correspondence (2.1) and the linearity of $d\rho(\cdot)$

we have

$$\forall \alpha_i : \quad \exp\big(\sum_i \alpha_i d\rho(A_i)\big)v = v.$$

Taking the derivative with respect to α_i at $\alpha = 0$, we get a constraint for each of the infinitesimal generators

$$\boxed{\forall i = 1, \ldots, D : \quad d\rho(A_i)v = 0} \tag{2.4}$$

For group elements with all $\alpha_i = 0$ and $N = 1$, we get an additional constraint for each of the discrete generators in the group:

$$\boxed{\forall k = 1, \ldots, M : \quad (\rho(h_k) - I)v = 0}. \tag{2.5}$$

We get a total of $O(M+D)$ constraints, one for each of the discrete and infinitesimal generators. In subsection A.1.2, we prove that these reduced constraints are not just **necessary** but also **sufficient**, and therefore characterize all solutions to the symmetry equation (2.3).

Solving the Constraint: We collect each of the symmetry constraints $C_1 = d\rho(A_1), C_2 = d\rho(A_2), \ldots, C_{D+1} = \rho(h_1) - I, \ldots$ into a single matrix C, which we can breakdown into its nullspace spanned by the columns of $Q \in \mathbb{R}^{m\times r}$ and orthogonal complement $P \in \mathbb{R}^{m\times(m-r)}$ using the singular value decomposition:

$$Cv = \begin{bmatrix} d\rho(A_1) \\ d\rho(A_2) \\ \ldots \\ \rho(h_1) - I \\ \ldots \end{bmatrix} v = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P^\top \\ Q^\top \end{bmatrix} v = 0. \tag{2.6}$$

All symmetric solutions for v must lie in the nullspace of C: $v = Q\beta$ for some coefficients β, and we can then parametrize all symmetric solutions directly in this subspace. Alternatively, defining

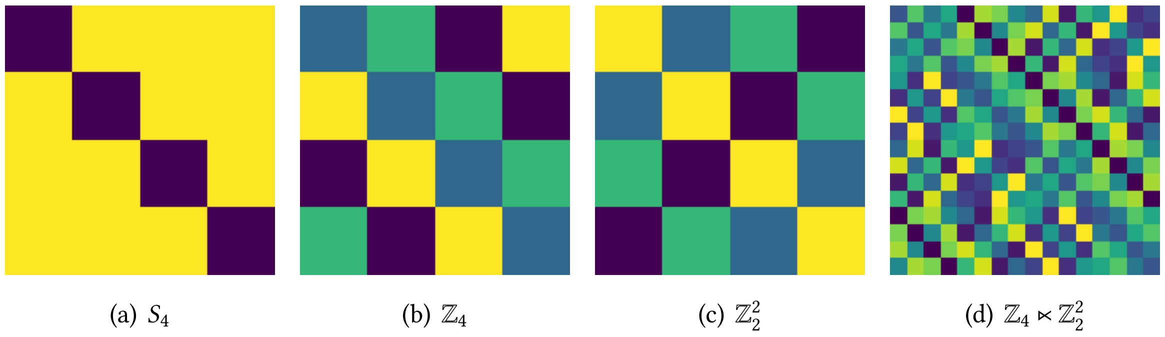


Figure 2.2: Equivariant basis for permutations, translation, 2d translation, and GCNN symmetries respectively, each of which are solutions to Equation 2.5 for different groups. The r different solutions in the basis are shown by different colors.

$\beta = Q^\top v_0$ we can reuse any standard parametrization and initialization, but simply project onto the equivariant subspace: $v = QQ^\top v_0$.

Thus given any finite dimensional linear representation, we can solve the constraints with a singular value decomposition.[1]. If $v \in \mathbb{R}^m$ the runtime of the approach is $O((M + D)m^3)$.

2.4.3 A Unifying Perspective on Equivariance

In order to make it more concrete and demonstrate its generality, we now show that standard convolutional layers [LeCun et al. 1989], deep sets [Zaheer et al. 2017], invariant graph networks [Maron et al. 2018], and GCNNs [Cohen and Welling 2016a] are examples of the solutions in equation (2.6) when specifying a specific symmetry group and representation.

Convolutions: To start off with the 1D case with sequences of n elements and a single channel, $V = \mathbb{R}^n$ is acted upon by cyclic translations from the group $G = \mathbb{Z}_n$. The group can be generated by a single element given by the permuation matrix $\rho(h) = \mathrm{P}[n, 1, 2, ..., n-1]$. Equivariant linear maps from $V \longrightarrow V$ are of type $T_{(1,1)}$. Expressing the representation and solving eq. (2.6) with SVD gives the $r = n$ matrices (reshaped from the rows of Q) shown by the circulant matrix in Figure 2.2, which is *precisely* the way to express convolution as a matrix.

[1]Equations (2.4) and (2.5) apply also to infinite dimensional representations, where ρ and $d\rho$ are linear operators acting on functions v, but solving these on a computer would be more difficult.

In the typical case of 2D arrays with $V = \mathbb{R}^{n^2}$ elements and multiple channels $c_{\text{in}}\ c_{\text{out}}$, there are $M = 2$ generators of the group $G = \mathbb{Z}_n \times \mathbb{Z}_n = \mathbb{Z}_n^2$ that are $\rho(h_1) = \rho(h) \otimes I$ and $\rho(h_2) = I \otimes \rho(h)$ defined in terms of the generator in the 1D case. For multiple channels, the mapping is $c_{\text{in}}V \to c_{\text{out}}V$ which has type $c_{\text{in}}c_{\text{out}}T_{(1,1)}$ which yields the matrix valued 2D convolution (with $c_{\text{in}}c_{\text{out}}n^2$ independent basis elements) that we are accustomed to using for computer vision[2].

Deep Sets: We can recover the solutions in Zaheer et al. [2017] by specifying $V = \mathbb{R}^n$ and considering S_n (permutation) equivariant linear maps $V \to V$. S_n can be generated in several ways such as with the $M = 2$ generators $\rho(h_1) = \mathrm{P}[1, n-1, 2, 3, ...]$ and $\rho(h_2) = \mathrm{P}[2, 1, 3, 4, ...]$ [Conrad 2013]. Solving the constraints for $T_{(1,1)}$ yields the $r = 2$ dimensional basis $Q = [I, \mathbb{1}\mathbb{1}^\top]$ shown in Figure 2.2.

Equivariant Graph Networks: Equivariant graph networks in Maron et al. [2018] generalize deep sets to S_n equivariant maps from $T_k \to T_\ell$, such as maps from adjacency matrices T_2 to themselves. They show these maps satisfy

$$\forall P \in S_n : \ P^{\otimes(k+\ell)}\mathrm{vec}(W) = \mathrm{vec}(W), \tag{2.7}$$

and use analytic techniques to find a basis, showing that the size of the basis is upper bounded[3] by the Bell numbers $1, 2, 5, 15, ...$. Noting that $P = (P^{-1})^\top$, we can now recognize $P^{\otimes k+\ell} = \rho_{(k,\ell)}(P)$ acting on the maps of type $T_{(k,\ell)}$. However we need not solve the combinatorially large Equation 2.7; our algorithm instead solves it just for the permutation generators $\rho(h_i)$, yielding the same solutions.

GCNNs: The Group Equivariant CNNs in Cohen and Welling [2016a] can be defined abstractly through fiber bundles and base spaces, but we can also describe them in our tensor notation. The original GCNNs have the $G = \mathbb{Z}_4 \ltimes (\mathbb{Z}_n \times \mathbb{Z}_n)$ symmetry group consisting of discrete translations of

[2]Note that the inductive bias of *locality* restricting from $n \times n$ filters to 3×3 filters is not a consequence of equivariance.

[3]For small n, the size of the equivariant basis for T_k can actually be less than B_k when $n^k < B_k$.

the grid, as well as 90^o rotations where $\ltimes$ is the semi-direct product. [4] In total, the representation space can be written $V = \mathbb{R}^4 \otimes \mathbb{R}^{n^2}$. We can now disentangle these two parts to read off the $M = 3$ generators for x, y translation and rotation. The translation generators are $I \otimes \rho(h_1)$ and $I \otimes \rho(h_2)$ from the 2D convolution section, as well a generator for rotation $P[4, 1, 2, 3] \otimes \mathrm{Rot}_{90}$ with the Rot_{90} matrix performing 90^o rotations on the grid. Solving for the constraint on $T_{(1,1)}$ yields the G-convolutional layer embedded in a dense matrix shown in Figure 2.2. Note the diagonal blocks implement rotated copies of a given filter, equivalent to the orientations in the regular representation of a GCNN.

Notably, each of these solutions for convolution, deep sets, equivariant graph networks, and GCNNs are produced as solutions from Equation 2.6 as a direct consequence of specifying the representation and the group generators. In subsection A.1.5 we calculate the equivariant basis for tensor representations of these groups $\mathbb{Z}_n$, S_n, D_n, as well as unexplored territory with $\mathrm{SO}(n)$, $\mathrm{O}(n)$, $\mathrm{Sp}(n)$, $\mathrm{SO}^+(1,3)$, $\mathrm{SO}(1,3)$, $\mathrm{O}(1,3)$, $\mathrm{SU}(n)$, and the Rubiks Cube group. We visualize several of these equivariant bases in Figure 2.3.

2.5 Efficiently Solving the Constraint

The practical application of our general approach is limited by two factors: the computational cost of computing the equivariant basis at initialization, and the computational cost of applying the equivariant maps in the forward pass of a network. In this section we address the scalability of the first factor, computing the equivariant basis.

The runtime for using SVD directly to compute the equivariant basis is too costly for all but very small representations $m = \dim(V) < 5000$. We improve upon the naive algorithm with two techniques: dividing the problem into a smaller set of independent subproblems and exploiting structure in the constraint matrices to enable an efficient iterative Krylov subspace approach for

[4] Cohen and Welling [2016a] also make D_4 dihedral equivariant networks that respect reflections, which can be accomodated by in our framework with 1 additional generator.

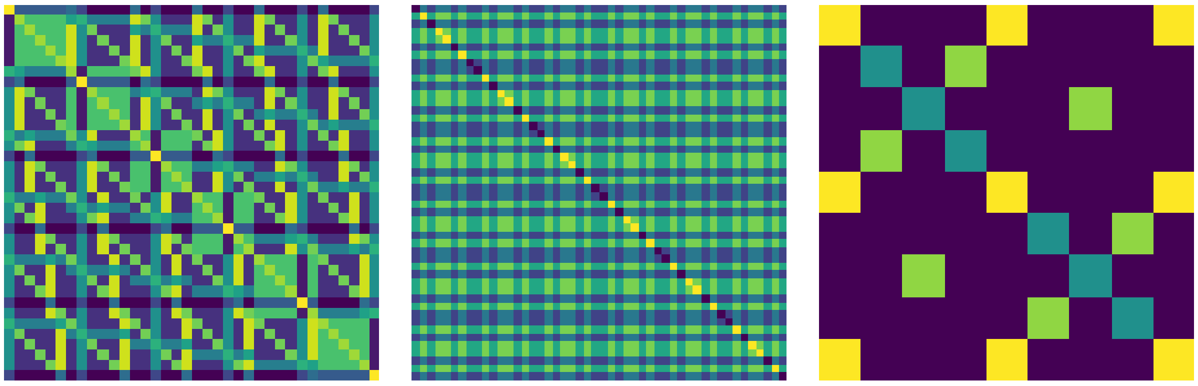

(a) $T_4^{S_6}$ (b) T_2^{Rubiks} (c) $T_4^{\text{SO}(3)}$

Figure 2.3: Equivariant basis for various tensor representations T_k^G where G denotes the symmetry group. The r different solutions in the basis are shown by different colors. For SO(3) the bases cannot be separated into disjoint set of 0 or 1 valued vectors, and so we choose overlapping colors randomly and add an additional color for 0.

computing the nullspace. These two techniques allow us to compute the bases for high dimensional representations while not sacrificing the equivariance or completeness of the solution basis. Our resulting networks run in time similar to a standard MLP.

2.5.1 Dividing into Independent Sub-problems

The feature space U in a neural network can be considered a combination of objects with different types and multiplicities. The features in standard CNN or deep set would be c copies of rank one tensors, $U = cT_1$, where c is the number of channels. Graph networks include both node features T_1 as well as edge features T_2 like the adjacency matrix. More general networks could have a mix of representations, for example 100 scalars, 30 vectors, 10 matrices and 3 higher order tensors: $U = 100T_0 \oplus 30T_1 \oplus 10T_2 \oplus 3T_3$. These composite representations with multiplicity are built from direct sums of simpler representations. $\rho_U(g) = \bigoplus_{a\in\mathcal{A}} \rho_a(g)$ for some collection of representations $\mathcal{A}$.

Since linear maps $U_1 \to U_2$ have the representation $\rho_2 \otimes \rho_1^*$, the product can be expanded as the direct sum

$$\rho_2 \otimes \rho_1^* = \bigoplus_{b\in\mathcal{A}_2} \rho_b \otimes \bigoplus_{a\in\mathcal{A}_1} \rho_a^* = \bigoplus_{(b,a)\in\mathcal{A}_2\times\mathcal{A}_1} \rho_b \otimes \rho_a^*. \tag{2.8}$$

Since $\oplus$ for both the group and algebra representations concatenates blocks along the diagonal, the constraints can be separated into the blocks given by each of the (b, a) pairs. Each of these constraints (2.4) and (2.5) can be solved independently for the $\rho_b \otimes \rho_a^*$ representation and then reassembled into the parts of the full matrix.

Unlike Steerable CNNs which use analytic solutions of irreducible representations [Cohen and Welling 2016b; Weiler and Cesa 2019a], we need not worry about any Clebsch-Gordon coefficients or otherwise, regardless of the representation used. [5] Note that tensor representations make things especially simple since $T_{(p,q)} \otimes T^*_{(r,s)} = T_{(p+s,q+r)}$, but are not required.

2.5.2 Krylov Method for Efficient Nullspaces

We can exploit structure in the matrices ρ and $d\rho$ for a more efficient solution. With this in mind, we propose to find the nullspace $Q \in \mathbb{R}^{n\times r}$ where r is the rank of the nullspace with the following optimization problem:

$$\min \; \|CQ\|_F^2 \quad s.t \quad Q^\top Q = I. \tag{2.9}$$

Minimizing using gradient descent, we have a very close relative of QR power iteration [Francis 1961] and Oja's rule [Garber and Hazan 2015; De Sa et al. 2015; Shamir 2015], that instead finds the smallest singular vectors. As the nullspace components are preserved by the gradient updates, the orthogonalization constraint can in fact be removed during the minimization and we list the steps of the iterative method in algorithm 1. Crucially, gradients require only matrix vector multiplies (MVMs) with the constraint matrix C, we never have to form the representation matrices explicitly and can instead implement an efficient MVM for ρ and $d\rho$. Through iterative doubling of the max

[5] For irreducible representations one typically decomposes $\rho_i \otimes \rho_j = Q^{-1}\big(\bigoplus_k \rho_k\big)Q$ with Clebsch-Gordan matrix Q, but we can leave the rep as $\rho_i \otimes \rho_j$ and solve numerically.

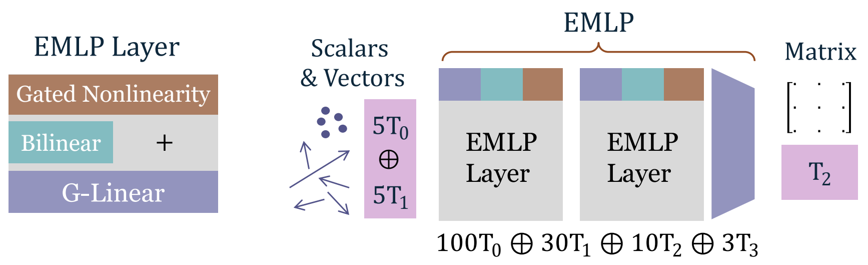


Figure 2.4: EMLP layers. G-equivariant linear layers, followed by the bilinear layer and a shortcut connection, and finally a gated nonlinearity. Stacking these layers together and choosing some internal representation (shown below), the EMLP maps some collection of geometric quantities to some other collection. Here we show the equivariant mappings from scalars and vectors to matrices.

rank r we need not know the true rank beforehand. As we prove in subsection A.1.3 the algorithm produces an ϵ accurate solution in time $O((M+D)\mathcal{T}r\log(1/\epsilon)+r^2n)$ where $\mathcal{T}$ is the time for an MVM with ρ and $d\rho$. The method is "exact" in the sense of numerical algorithms in that we can specify a precision ϵ close to machine precision and converge in $\log(1/\epsilon)$ iterations due to the exponential convergence rate, which we verify in Figure A.1.

Algorithm 1 Fast Krylov Nullspace

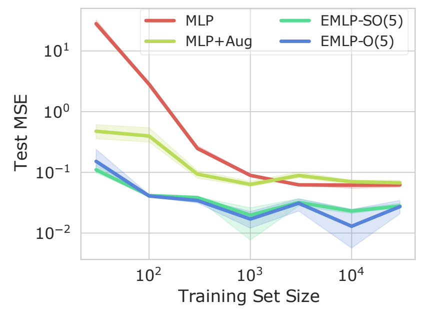
```
def KrylovNullspace($C$):
$r_{\max} = r = 10$
while $r = r_{\max}$ do
    $r_{\max} \leftarrow 2r_{\max}$
    $Q = \mathrm{CappedKrylovNullspace}(C, r_{\max})$
    $r \leftarrow \mathrm{rank}(Q)$
end
return $Q$
def CappedKrylovNullspace($C, r_{\max}$):
$Q \sim \mathcal{N}(0,1)^{n\times r_{\max}}$
while $L(Q) > \epsilon$ do
    $L(Q) = \|CQ\|_F^2$
    $Q \leftarrow Q - \eta\nabla L$
end
$Q, \Sigma, V = \mathrm{SVD}(Q)$
return $Q$
```

The pairs of tensor products of representations, $\rho_b(h)\otimes\rho_a(h^{-1})^\top$ and $d\rho_b(A)\bar{\oplus}(-d\rho_a(A)^\top)$ from Equation 2.8 have Kronecker structure allowing efficent MVMs $(A\otimes B)\mathrm{vec}(W) = \mathrm{vec}(AWB^\top)$.

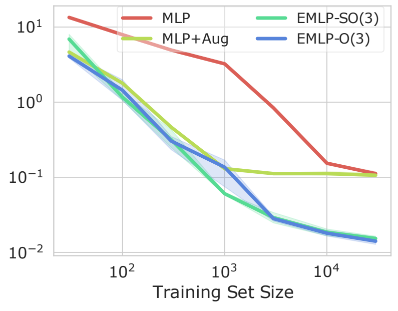
Exploiting this structure alone, solving the constraints for a matrix $W \in \mathbb{R}^c \to \mathbb{R}^c$ takes time

$$O((M + D)\mathcal{T}rc + r^2c^2) \tag{2.10}$$

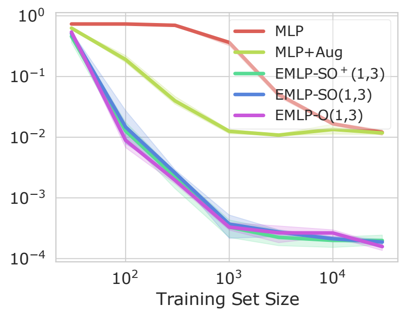
where $\mathcal{T}$ is the time for MVMs with constituent matrices $\rho_a, \rho_b, d\rho_a, d\rho_b$. For some of the groups this time $\mathcal{T}$ is in fact a *constant*, for example the permutation generators merely swap two entries, and Lie algebras can often be written in a sparse basis. For high order tensor representations, one can exploit higher order Kronecker structure. Even for discrete groups the runtime is a strict improvement over the approach by van der Pol et al. [2020b] which runs in time $O(|G|\mathcal{T}rc + r^2c^2)$. For large discrete groups like S_n our approach gives an exponential speedup, $O(n!) \to O(n)$.



(a) O(5) invariant Synthetic



(b) O(3) equivariant Inertia



(c) O(1,3) invariant $(e^- + \mu^-)$-Scattering

Figure 2.5: Data efficiency for the synthetic equivariance experiments. Here the EMLP-G models where G are relevant symmetry groups strongly outperforms both standard MLPs and MLPs that have been trained with data augmentation to the given symmetry group, across the range of dataset sizes. The shaded regions depict 95% confidence intervals taken over 3 runs.

2.6 Network Architecture

While the constraint solving procedure can be applied to any linear representations, we will use tensor representations to construct our network. The features in each layer are a collection of tensors of different ranks $v \in U = \bigoplus_{a\in\mathcal{A}} T_{(p_a,q_a)}$ with the individual objects $v_a \in T_{(p_a,q_a)}$. As a heuristic, we allocate the channels uniformly between tensor ranks for the intermediate layers.

For example with 256 channels for an SO(3) equivariant layer with $\dim(T_{(p,q)}) = 3^{p+q}$, uniformly allocating channels produces $U = 70T_0 \oplus 23T_1 \oplus 7T_2 \oplus 2T_3$. The input and output layers are set by the types of the data. To build a full equivariant multilayer perceptron (EMLP) from the equivariant linear layer, we also need equivariant nonlinearities.

Gated Nonlinearities: For this purpose we use *gated* nonlinearities introduced in Weiler et al. [2018]. Gated nonlinearities act separately for each of the different objects in the features (that are concatenated through the direct sum $z = \mathrm{Concat}(\{v_a\}_{a\in\mathcal{A}})$). The nonlinearity takes values $\mathrm{Gated}(v_a) = v_a\sigma(s_a)$ where s_a is a scalar 'gate' for each of the objects. For scalar objects $v_a \in T_0 = \mathbb{R}^1$ and regular representations (which allow pointwise nonlinearities), the gate is just the object itself and so the nonlinearity is just Swish [Ramachandran et al. 2017]. For other representations the gate scalars are produced as an additional output of the previous layer.

Universality: The theorem in Maron et al. [2019] shows that tensor networks with pointwise nonlinearities and G-equivariant linear layers for $G \leqslant S_n$ are universal. However, this result does not extend to the gated nonlinearities required for other groups and representations. As we prove in subsection A.1.4, gated nonlinearities are *not* sufficient for universality in this general case, and can be extremely limiting in practice. The problem relates to not being able to express any kind of contractions between elements with the different objects within a feature layer (like a dot product).

Cheap Bilinear Layers. To address this limitation we introduce an inexpensive bilinear layer which performs tensor contractions on pairs of input objects that produce a given output type. Explicitly, two input objects $v_a \in T_{(a_1,a_2)}$ and $v_b \in T_{(b_1,b_2)}$ can be contracted to give a type $T_{(c_1,c_2)}$ if and only if $(a_1, a_2) = (c_1 + b_2, c_2 + b_1)$ or $(b_1, b_2) = (c_1 + a_2, c_2 + a_1)$. In other words, if v_a can be interpreted as a linear map from $T_b \longrightarrow T_c$ then we can apply $y_c = \mathrm{Reshape}(v_a)v_b$ and vice versa. We add a learnable parameter weighting each of these contractions (excluding scalars).

We can now assemble the components to build a full equivariant multilayer perceptron (EMLP) from the equivariant linear layer, the gated nonlinearities, and the additional bilinear layer. We show how these components are assembled in Figure 2.4.

2.7 Experiments

We evaluate EMLP on several synthetic datasets to test its capability on previously unexplored groups, and apply our model to the task of learning dynamical systems with symmetry.

2.7.1 Synthetic Experiments

O(5) Invariant Task: To start off, we evaluate our model on a synthetic O(5) invariant regression problem $2T_1 \longrightarrow T_0$ in $d = 5$ dimensions given by the function $f(x_1, x_2) = \sin(\|x_1\|) - \|x_2\|^3/2 + \frac{x_1^\top x_2}{\|x_1\|\|x_2\|}$. We evaluate EMLP-SO(5) and EMLP-O(5) which is also equivariant to reflections. We compare against a standard MLP as well as MLP-Aug that is trained with O(5) data augmentation. We show the results in Figure 2.5.

O(3) **Equivariant Task**: Next we evaluate the networks on the equivariant task of predicting the moment of inertia matrix $\mathcal{I} = \sum_i m_i(x_i^\top x_i I - x_i x_i^\top)$ from $n = 5$ point masses and positions. The inputs $X = \{(m_i, x_i)\}_{i=1}^5$ are of type $5T_0 + 5T_1$ (5 scalars and vectors) and outputs are of type T_2 (a matrix), both transform under the group. We apply SO(3) and O(3) equivariant models to this problem. For the baselines, we implement data augmentation for the standard MLP for this equivariant task by simultaneously transforming the input by a random matrix $R \in \mathrm{O}(3)$ and transforming the output accordingly by the inverse transformation: $\hat{y} = R^\top \mathrm{MLP}(\{(m_i, Rx_i\}_{i=1}^5)R$. This kind of equivariant data augmentation that transforms both the input and the output according to the symmetry is a strong baseline.

Lorentz Equivariant Particle Scattering: Testing the ability of the model to handle Lorentz equivariance in tasks relevant to particle physics, we train models to fit the matrix element in electron muon scattering $e^- + \mu^- \longrightarrow e^- + \mu^-$ which is proportional to the scattering cross-section.

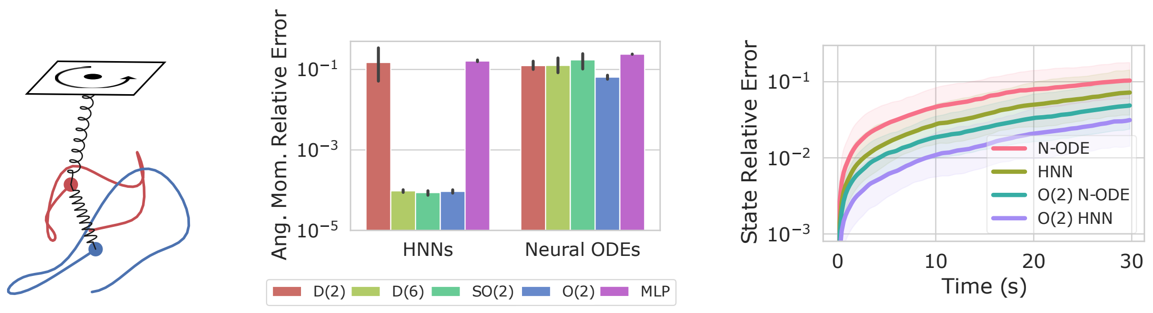


Figure 2.6: Left: A double spring pendulum (12s sample trajectory is shown). The system has an $O(2)$ symmetry about the z axis. **Middle**: Conservation of angular momentum about the z-axis (the geometric mean of the relative error is computed over $30s$ rollouts and averaged across initial conditions). Errorbars are 95% confidence interval over 3 runs. **Right**: The relative error in the state as the trajectory is rolled out. Shaded regions show 1 standard deviation in log space across the different trajectories rather than models, showing the variance in the data.

The scattering matrix element is proportional to $|\mathcal{M}|^2 \propto$

$$[p^{(\mu}\tilde{p}^{\nu)} - (p^\alpha\tilde{p}_\alpha - p^\alpha p_\alpha)\eta^{\mu\nu}][q_{(\mu}\tilde{q}_{\nu)} - (q^\alpha\tilde{q}_\alpha - q^\alpha q_\alpha)\eta_{\mu\nu}]$$

[Martin 2012] where q_μ and p_μ are the four momenta for the ingoing electron and muon respectively, while $\tilde{q}_\mu$ and $\tilde{p}_\mu$ are the outgoing momenta, and parentheses $(\mu\nu)$ denotes the symmetrization of indices and repeated indices are contracted. While simple enough express in closed form, the scalar output involves contractions, symmetrization, upper and lower indices, and a metric tensor. Here the inputs are $4T_{(1,0)}$ and the output is a scalar $T_{(0,0)}$. We evaluate EMLP with equivariance not just to the proper orthochronious Lorentz group $SO^+(1,3)$ from Bogatskiy et al. [2020], but also the special Lorentz group $SO(1,3)$, and the full Lorentz group $O(1,3)$ and compare a MLP baseline that uses $O(1,3)$ data augmentation.

As shown in Figure 2.5, our EMLP model with the given equivariance consistently outperform sthe baseline MLP trained with and without data augmentation across the different dataset sizes and tasks, often by orders of magnitude.

2.7.2 Modeling dynamical systems with symmetries

Finally we turn to the task of modeling dynamical systems. For dynamical systems, the equations of motion can be written in terms of the state $\mathbf{z} \in \mathbb{R}^m$ and time t as $d\,\mathbf{z}/dt = F(\mathbf{z}, t)$. Neural ODEs [Chen et al. 2018] provide a way of learning these dynamics directly from trajectory data. A neural network parametrizes the function F_θ and the learned dynamics can be rolled out using a differentiable ODE solver $(\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_T) = \mathrm{ODESolve}(\mathbf{z}_0, F_\theta, (t_1, t_2, ..., t_T))$ and fit to trajectory data with the L2 loss $L(\theta) = \frac{1}{T}\sum_{t=1}^{T} ||\hat{\mathbf{z}}_t - \mathbf{z}_t||_2^2$.

Many physically occurring systems have a 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 in terms of the gradients of a scalar $\mathcal{H}(z)$ known as the Hamiltonian, which often coincides with the total energy. $\frac{d\mathbf{z}}{dt} = J\nabla\mathcal{H}$ with $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$. As shown in Greydanus et al. [2019] with Hamiltonian Neural Networks (HNNs), one can exploit this Hamiltonian structure by parametrizing $\hat{\mathcal{H}}_\theta(\mathbf{z})$ with a neural network, and then taking derivatives to find the implied Hamiltonian dynamics. For problems with Hamiltonian structure HNNs often lead to improved performance, and better energy conservation.

A dynamical system can have *symmetries* such as the symmetries given by $F(\rho(g)\,\mathbf{z}, t) = \rho(g)F(\mathbf{z}, t)$ for some linear representation, which is equivariance in the first argument. Meanwhile Hamiltonian dynamics have symmetries according to invariances of the Hamiltonian $\mathcal{H}(\rho(g)\,\mathbf{z}) = \mathcal{H}(\mathbf{z})$. Continuous symmetries of the Hamiltonian are of special significance since they produce conservation laws such as conservation of linear and angular momentum or conservation of charge as part of the Noether theorem [Noether 1971].

We apply our EMLP model to the task of learning the dynamics of a double pendulum connected by springs in 3D shown in Figure 2.6. The problem exhibits a O(2) rotational and reflectional symmetry about the z-axis as well as Hamiltonian structure. As the state space cannot be traversed

	O(2)	SO(2)	D_6	MLP
N-ODEs:	**0.019(1)**	0.051(36)	0.036(25)	0.048
HNNs:	**0.012(2)**	0.015(3)	0.013(2)	0.028

Table 2.2: Geometric mean of rollout errors (relative error) over T=30s for the various EMLP-G symmetric HNNs and Neural ODEs (N-ODE) vs ordinary MLP HNNs and N-ODEs. Errorbars are 1 standard deviation computed over 3 trials, with notation .012(2) meaning .012 ± .002.

by the group elements alone, it is not a homogeneous space, a setting that has been explored very little in the equivariance literature [Cohen et al. 2018a].

However, we can readily use EMLP on this problem and we show in Table 2.2 and Figure 2.6 that exploiting the O(2) symmetry (and subgroups SO(2), D_6) with EMLP leads to improved performance for both Neural ODE and HNN models. Furthermore, enforcing the continuous rotation symmetry in the EMLP-HNN models yields conservation of angular momentum about the z-axis, a useful property for learned simulations. Interestingly the dihedral group D_6 which is discrete does not satisfy Noether's theorem and yet it still yields approximate angular momentum conservation, but the coarser D_2 symmetry does not. As expected, all Neural ODE models do not conserve angular momentum as Noether's theorem only applies to the Hamiltonians and not to the more general ODEs. While conservation laws from learning invariant Hamiltonians was also explored in Finzi et al. [2020] with LieConv, LieConv models assume permutation equivariance which is broken by the pivot in this system. Because EMLP is general, we can apply it to this non permutation symmetric and non transitively acting rotation group that is embedded in the larger state space.

2.8 Discussion

We presented a construction for equivariant linear layers that is completely general to the choice of representation and matrix group. Convolutions, deep sets, equivariant graph networks and GCNNs all fall out of the algorithm naturally as solutions for a given group and representation. Through an iterative MVM based approach, we can solve for the equivariant bases of very large representations. Translating these capabilities into practice, we build EMLP and apply the model to problems with symmetry including Lorentz invariant particle scattering and dynamical systems, showing consistently improved generalization.

Though EMLP is not much slower than a standard MLP, dense matrix multiplies in an MLP and our EMLP make it slow to train models the size of convnets or large graph networks which have specialized implementations. With the right techniques, this apparent generality-specialization tradeoff may be overcome. The flexibility of our approach should lower the costs of experimentation and allow researchers to more easily test out novel representations. Additionally we hope that our constraint solver can help launch a variety of new methods for learning symmetries, modeling heterogeneous data, or capturing prior knowledge.

3 | Residual Pathway Priors for Approximate Equivariance

There is often a trade-off between building deep learning systems that are expressive enough to capture the nuances of the reality, and having the right inductive biases for efficient learning. We introduce Residual Pathway Priors (RPPs) as a method for converting hard architectural constraints into soft priors, guiding models towards structured solutions, while retaining the ability to capture additional complexity. Using RPPs, we construct neural network priors with inductive biases for equivariances, but without limiting flexibility. We show that RPPs are resilient to approximate or misspecified symmetries, and are as effective as fully constrained models even when symmetries are exact. We showcase the broad applicability of RPPs with dynamical systems, tabular data, and reinforcement learning. In Mujoco locomotion tasks, where contact forces and directional rewards violate strict equivariance assumptions, the RPP outperforms baseline model-free RL agents, and also improves the learned transition models for model-based RL.

This chapter is adapted from the paper “Residual Pathway Priors for Soft Equivariance Constraints”, which originally appeared at NeurIPS 2022 and is joint work with Greg Benton, and Andrew Gordon Wilson.

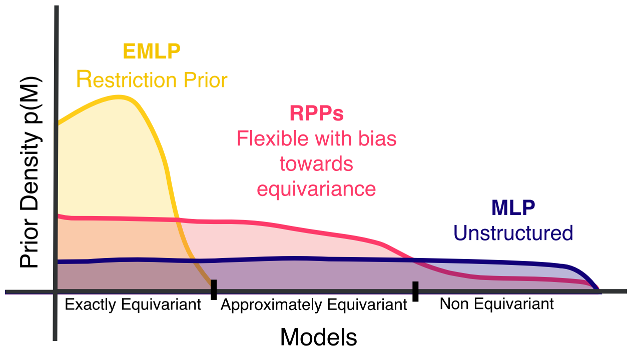
3.1 Introduction

Central to the expanding application of deep learning to structured data like images, text, audio, sets, graphs, point clouds, and dynamical systems, has been a search for finding the appropriate set of inductive biases to match the model to the data. These inductive biases, such as recurrence [Rumelhart et al. 1985], local connectivity [LeCun et al. 1989], equivariance [Cohen and Welling 2016a], or differential equations [Chen et al. 2018], reduce the set of explored hypotheses and improve generalization. Equivariance in particular has had a large impact as it allows ruling out a large class of meaningless shortcut features in many distinct domains, such as the ordering of the nodes in graphs and sets or the coordinate system chosen for an image.

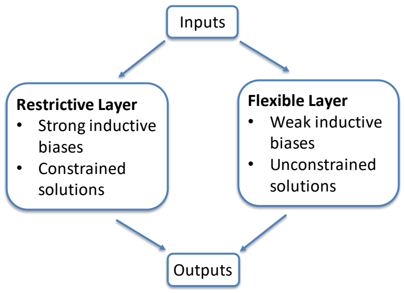
A disadvantage of hard coding these restrictions is that this prior knowledge may not match reality. A scene may have long range non-local interactions, rotation equivariance may be violated by a preferred camera angle, or a dynamical system may occasionally have discontinuous transitions. In particular, symmetries are delicate. A small perturbation like adding wind breaks the rotational symmetry of a pendulum, and bumpy or tilted terrain could break the translation symmetry for locomotion. In these cases we would like to incorporate our prior knowledge in a way that admits our own ignorance, and allows for the possibility that the world is more complex than we imagined. We aim to develop an approach that is more general, and can be applied when symmetries are exact, approximate, or non-existent.

The Bayesian framework provides a mechanism for expressing such knowledge through priors. In much of the past work on Bayesian neural networks, the relationship between the prior distribution and the functions preferred by the prior is not transparent. While it is easy to specify different variances for different channels, or to use heavy tailed distributions, it is hard know how high level properties meaningfully translate into these low level attributes. Ultimately priors should represent our prior *beliefs*, and the beliefs we have are about high level concepts like the

*Equal Contribution



(a) Priors over Equivariant Solutions



(b) Structure of RPP Models

Figure 3.1: Left: RPPs encode an Occam's razor approach to modeling. Highly flexible models like MLPs lack the inductive biases to assign high prior mass to relevant solutions for a given problem, while models with strict constraints are not flexible enough to support solutions with only approximate symmetry. For a given problem, we want to use the most constrained model that is consistent with our observations. **Right:** The structure of RPPs. Expanding the layers into a sum of the constrained and unconstrained solutions, while setting the prior to favor the constrained solution, leads to the more flexible layer explaining only the *residual* of what is already explained by the constrained layer.

locality, independence, and symmetries of the data.

To address the need for more interpretable priors we introduce *Residual Pathway Priors* (RPPs), a method for converting hard architectural constraints into soft priors. Practically, RPPs allow us to tackle problems in which perfect symmetry has been violated, but approximate symmetry is still present, as is the case for most real world physical systems. RPPs have a prior bias towards equivariant solutions, but are not constrained to them.

We use the schematic in Figure 3.1 (left) as an approach to model construction [Wilson and Izmailov 2020; MacKay and Mac Kay 2003]. The flexibility of our model is described by what solutions have non-zero prior probability density. The *inductive biases* are described by the distribution of support over solutions. We wish to construct models with inductive biases that assign significant prior mass for solutions we believe to be a priori likely, but without ruling out other solutions we believe to be possible. For example, models constrained to exact symmetries could not fully represent many problems, such as the motion of a pendulum in the presence of wind. Flexible models with poor inductive biases, spread thinly across possible solutions, could express an approximate symmetry, but such solutions are unlikely to be found because of the low prior density. In this sense, we wish to embrace a notion of Occam's razor such that "everything should be made as simple as possible, but no simpler".

As we find with problems in which symmetries exist, highly flexible models with weak inductive biases like MLPs fail to concentrate prior mass around solutions that exhibit any symmetry. On the other hand when symmetries are only approximate, the strong restriction biases of constrained models like Equivariant Multi-Layer Perceptrons (EMLP) [Finzi et al. 2021] fail to provide support for the observations. As a middle ground between these two extremes, RPPs combine the inductive biases of constrained models with the flexibility of MLPs to define a model class which excels when data show approximate symmetries, as shown in Figure 3.1 (right).

In the following sections we introduce our method and show results across a variety of domains. We list our contributions and the accompanying sections below:

1. We propose *Residual Pathway Priors* as a mechanism to imbue models with soft inductive biases, without constraining flexibility.
2. While our approach is general, we use RPPs to show how to turn hard architectural constraints into soft equivariance priors (Section 3.4).
3. We demonstrate that RPPs are robust to varying degrees of symmetry (Section 3.5). RPPs perform well under exact, approximate, or misspecified symmetries.
4. Using RPP on the approximate symmetries in the complex state spaces of the Mujoco locomotion tasks, we improve the performance of model free RL agents (Section 3.6).

We provide a PyTorch implementation of residual pathway priors at `https://github.com/mfinzi/residual-pathway-priors`.

3.2 Related Work

The challenge of equivariant models not being able to fully fit the data has been identified in a number of different contexts, and with different application specific adjustments to mitigate the problem. Liu et al. [2018] observe that convolutional networks can be extremely poor at tasks that require identifying or outputting spatial locations in an image as a result of the translation symmetry. The authors solve the problem by concatenating a coordinate grid to the input of the convolution layer. Constructing translation and rotation equivariant GCNNs, Weiler and Cesa [2019a] find that in order to get the best performance on CIFAR-10 and STL-10 datasets which have a preferred camera orientation, they must break the symmetry, which they do by using equivariance to progressively smaller subgroups in the later layers. Bogatskiy et al. [2020] go to great lengths to construct Lorentz group equivariant networks for tagging collisions in particle colliders only to break the symmetry by introducing dummy inputs that identify the collision axis. van der Wilk et al. [2018] use the marginal likelihood to learn approximate invariances in Gaussian processes from data. In a related procedure, Benton et al. [2020] learn the *extent* of symmetries in neural networks using the reparametrization trick and test time augmentation. While sharing some commonalities with RPP, this method is not aimed at achieving approximate equivariance and cannot bake equivariance into the model architecture.

A separate line of work has attempted to combine the extreme flexibility of the Vision Transformer (ViT) [Dosovitskiy et al. 2020] with the better sample efficiency of convolutional networks, by incorporating convolutions at early layers [Xiao et al. 2021] or making the self attention layer more like a convolution [d'Ascoli et al. 2021; Dai et al. 2021]. Most similar to our work, ConViT [d'Ascoli et al. 2021] uses a gating mechanism for adding a soft locality bias to the self attention mechanism in Vision Transformers. ConViT and RPP share the same motivation, but while ConViT is designed specifically for biasing towards locality in the self attention layer, RPP is a general approach that we can apply broadly with other kinds of layers, symmetries, or architectural constraints.

Outside of equivariance, adding model outputs to a much more restrictive base model has been a fruitful idea employed in multiple contexts. The original ResNet [He et al. 2016a,c] drew on this motivation, with shortcut connections. Johannink et al. [2019] and Silver et al. [2018] proposed Residual Reinforcement Learning, whereby the RL problem is split into a user designed controller using engineering principles and a flexible neural network policy learned with RL. Similarly, in modeling dynamical systems, one approach is to incorporate a base parametric form informed by models from physics or biology, and only learn a neural network to fit the delta between the simple model and reality [Kashinath et al. 2021; Liu et al. 2021b].

There have been several works tackling symmetries and equivariance in RL, such as permutation equivariance for multi-agent RL [Sukhbaatar et al. 2016; Jiang et al. 2018; Liu et al. 2020], as well exploring reflection symmetry for continuous control tasks [Abdolhosseini et al. 2019], and discrete symmetries in the more general framework of MDP homomorphisms [van der Pol et al. 2020b]. However, in each of these applications the symmetries need to be exact, and the complexities of real data often require violating those symmetries. Although not constructed with this purpose, some methods which use regularizers to enforce equivariance [van der Pol et al. 2020a] could be used for approximate symmetries. Interestingly, the value of approximate symmetries of MDPs has been explored in some theoretical work [Ravindran and Barto 2004; Taylor et al. 2008], but without architectures that can make use of it. Additionally, data augmentation, while not able to bake in architectural equivariance, has been successfully applied to encouraging equivariance on image tasks [Kostrikov et al. 2020] and recently even on tabular state vectors [Lin et al. 2020; Mavalankar 2020].

3.3 Background

In order develop our method, we first review the concept of group symmetries, how representations formalize the way these symmetries act on different objects.

Group Symmetries In the machine learning context, a symmetry group G can be understood as a set of invertible transformations under which an object is the same, such as reflections or rotations. These symmetries can act on many different kinds of objects. A rotation could act on a simple vector, a 2d array like an image, a complex collection objects like the state space of a robot, or more abstractly on an entire classification problem or Markov Decision Process (MDP).

Representations The way that symmetries act on objects is described by a *representation.* Given an object in an n-dimensional vector space V, a group representation is a mapping $\rho : G \longrightarrow \mathbb{R}^{n\times n}$, yielding a matrix which acts on V. Vectors $v \in V$ are transformed $v \mapsto \rho(g)v$. In deep learning, each of the inputs and outputs to our models can be embedded in some vector space: an $m \times m$ sized rgb image exists in $\mathbb{R}^{3m^2}$, and a node valued function on a graph of m elements exists within $\mathbb{R}^m$. The representation ρ specifies how each of these objects transform under the symmetry group G.

These representations can be composed of multiple simpler subrepresentations, describing how each object within a collection transforms. For example given the representation ρ_1 of rotations acting on a vector in $\mathbb{R}^3$, and a representation ρ_2 of how rotations act on a 3×3 matrix, the two objects concatenated together have a representation given by $\rho_1(g) \oplus \rho_2(g) = \begin{bmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{bmatrix}$, where the two matrices are concatenated along the diagonal. Practically this means we can represent intricate and multifaceted structures by breaking them down into their component parts and defining how each part transforms. For example, we may know that the velocity vector, an orientation quaternion, a joint angle, and a control torque all transform in different ways under a left-right reflection, and one can accommodate this information into the representation.

Equivariance Given some data X with representation ρ_{in}, and Y with representation ρ_{out}, we may wish to learn some mapping $f : X \to Y$. A model f is equivariant [Cohen and Welling 2016a], if applying the symmetry transformation to the input is equivalent to applying it to the output

$$f(\rho_{\text{in}}(g)x) = \rho_{\text{out}}(g)f(x).$$

In other words, it is not the symmetry of X or Y that is relevant, but the symmetry of the function f mapping from X to Y. If the true relationship in the data has a symmetry, then constraining the hypothesis space to functions f that also have the symmetry makes learning easier and improves generalization [Elesedy and Zaidi 2021]. Equivariant models have been developed for a wide variety of symmetries and data types like images [Cohen and Welling 2016a; Worrall et al. 2017; Zhou et al. 2017; Weiler and Cesa 2019a], sets [Zaheer et al. 2017; Maron et al. 2020], graphs [Maron et al. 2018], point clouds [Anderson et al. 2019; Fuchs et al. 2020; Satorras et al. 2021], dynamical systems [Finzi et al. 2020], jets [Bogatskiy et al. 2020], and other objects [Wang et al. 2020; Finzi et al. 2021].

3.4 Residual Pathway Priors

In this section, we introduce Residual Pathway Priors (RPPs). The core implementation of the RPP is to expand each layer in model into a sum of both a restrictive layer that encodes the hard architectural constraints and a generic more flexible layer, but penalize the more flexible path via a lower prior probability. Through the difference in prior probability, explanations of the data using only the constrained solutions are prioritized by the model; however, if the data are more complex the residual between the target and the constrained layer will be explained using the flexible layer. We can apply this procedure to any restriction priors, such as linearity, locality, Markovian structure, and, of course, equivariance.

Provided we can represent the r dimensional orthogonal basis of the k weights (A) in a constrained model as $Q \in \mathbb{R}^{k\times r}$, then we can define a Gaussian prior over the weights in that basis as $A \sim \mathcal{N}(0, \sigma_a^2 QQ^\top)$. Since Q is orthogonal we can define it's orthogonal complement as P, then a Gaussian prior over unconstrained weights (B) can be written $B \sim \mathcal{N}(0, \sigma_b^2 I) = \mathcal{N}(0, \sigma_b^2 QQ^\top + \sigma_b^2 PP^\top)$. Thus the prior over the sum of the weights of the constrained and unconstrained layers is

$$A + B \sim \mathcal{N}(0, (\sigma_a^2 + \sigma_b^2)QQ^T + \sigma_b^2 PP^T). \tag{3.1}$$

Regardless of the values of the prior variances σ_a^2 and σ_b^2, solutions in the constrained subspace $QQ^\top$ are automatically favored by the model and assigned higher prior probability mass than those in the subspace $PP^\top$ that violate the constraint. Even if $\sigma_b > \sigma_a$, the model still favors equivariance because the equivariance solutions are contained in the more flexible layer A. We show in Section **??** that RPPs are insensitive to the choice of σ_a and σ_b, provided that σ_a is large enough to be able to fit the data.

The Residual Pathway Prior draws inspiration from the residual connections in ResNets [He et al. 2016a,c], whereby training stability and generalization improves by providing multiple paths for gradients to flow through the network that have different properties. One way of interpreting a residual block and shortcut connection $f(x) = x + h(x)$ in combination with l2 regularization, either explicitly from weight decay or implicitly from the training dynamics [Neyshabur et al. 2014], is as a prior that places higher prior likelihood on the much simpler identity mapping than on the more flexible function $h(x)$. In this way, $h(x)$ need only explain the the difference between what is explained in the previous layer (passed through by I) and the target.

Under the prior of Equation 3.2, a MAP optimized model will favor explanations of the data using the more structured layer A, and only resort to using layer B to explain the *difference* between the target and what is already explained by the more structured model A. Adding these unconstrained residual pathways to each layer of an constrained model, we have a model that has the same expressivity of a network formed entirely of B layers, but with the inductive bias towards a model formed entirely with the constrained A layers. We term this model a *Residual Pathway Prior.*

To make the approach concrete, we first consider constructing equivariance priors using the constraint solving approach known as Equivariant Multi-Layer Perceptrons (EMLP) from Finzi et al. [2021].

Equivariant MLPs EMLPs provide a method for automatically constructing exactly equivariant layers for any given group and representation by solving a set of constraints. The way in which the vectors are equivariant is given by a formal specification of the types of the input and output through defining their representations. Given some input vector space V_{in} with representation ρ_{in} and some output space V_{out} with representation ρ_{out} the space of all equivariant linear layers mapping $V_{\text{in}} \to V_{\text{out}}$ satisfies

$$\forall g \in G: \quad \rho_{\text{out}}(g)W = W\rho_{\text{in}}(g).$$

These solutions to the constraint form a subspace of matrices $\mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$ which can be solved for and described by a r dimensional orthonormal basis $Q \in \mathbb{R}^{n_{\text{out}} n_{\text{in}} \times r}$. Linear layers can then be parametrized in this equivariant basis. The elements of W can be parametrized $\text{vec}(W) = Q\beta$ for $\beta \in \mathbb{R}^r$ for the linear layer $v \mapsto Wv$, and symmetric biases can be parametrized similarly.

Equivariance Priors with EMLP In order to convert the hard equivariance constraints in EMLP into a soft prior over equivariance that can accommodate approximate symmetries, we can apply the RPP procedure from above to each these linear layers in the network. Instead of parametrizing the weights W directly in the equivariant basis $\text{vec}(W) = Q\beta$, we can instead define W as the sum $W = A + B$ of an equivariant weight matrix $\text{vec}(A) = Q\beta$ an unconstrained weight matrix B. Placing Gaussian priors over both A and B yields the RPP prior in Equation (3.1) with $A + B = W \sim \mathcal{N}(0, (\sigma_a^2 + \sigma_b^2)QQ^\top + \sigma_b^2 PP^\top)$.

By replacing each of the equivariant linear layers in an EMLP with a sum of an equivariant layer and an unconstrained layer and adding in the negative prior likelihood to the loss function, we produce an RPP-EMLP that can accommodate approximate or incorrectly specified symmetries. [1]

RPPs With Other Equivariant Models While in EMLP equivariant bases are solved for explicitly, the RPP can be applied to the linear layers in other equivariant networks in precisely the same way. A good example is the translationally equivariant convolutional neural network (CNN), which can be viewed as a restricted subset of a fully connected network. Though the layers are parametrized as convolutions, the convolution operation can be expressed as a Toeplitz matrix residing within the space of dense matrices. Adding the convolution to a fully connected layer and choosing a prior variance σ_a^2 and σ_b^2 over each, we have the same RPP prior

$$W \sim N(0, \sigma_a^2 QQ^\top + \sigma_b^2 I) \tag{3.2}$$

where Q is the basis of (bi-)Toeplitz matrices corresponding to 3×3 filters. This RPP CNN has the biases of convolution but can readily fit non translationally equivariant data. We can similarly create priors with the biases of other equivariant models like GCNNs [Cohen and Welling 2016a], without any hard constraints. We can even apply the RPP principle to the breaking of a given symmetry group to a subgroup.

[1] For the EMLP that uses gated nonlinearities which do not always reduce to a standard Swish, we likewise add a more general Swish weighted by a parameter with prior variance σ_b^2.

3.5 How and Why RPPs Work

We explore how and why RPPs work on a variety of domains, applying RPPs where (1) constraints are known to be helpful, (2) cannot fully describe the problem, and (3) are misspecified.

3.5.1 Dynamical Systems and Levels of Equivariance

In order to better understand how and why residual pathway priors interact with the symmetries of the problem we move to settings in which we can directly control both the type of symmetry and the level to which the symmetries are violated. We examine how RPPs coupled with EMLP networks (RPP-EMLP) perform on the inertia and double pendulum datasets featured in Finzi et al. [2021] in 3 experimental settings: (i) the original inertia and double pendulum datasets which preserve exact symmetries with respect to the to $\mathrm{O}(3)$ and $\mathrm{O}(2)$ groups respectively; (ii) modified versions of these datasets with additional factors (such as wind on the double pendulum) that lead to approximate symmetries; and (iii) versions with misspecified symmetry groups that break the symmetries entirely (described in subsection A.2.3).

The results for these 3 settings are given in Figure 3.2. Across all settings RPP-EMLP match the performance of EMLP when symmetries are exact, perform as well as an MLP when the symmetry is misspecified and better than both when the symmetry is approximate. For these experiments we use a prior variance of $\sigma_a^2 = 10^5$ on the EMLP weights and $\sigma_b^2 = 1$ on the MLP weights.

Exact Symmetries As part of the motivation, RPPs should properly allocate prior mass to both constrained and unconstrained solutions, we test cases in which symmetries are exact, and show that RPP-EMLP is capable of performing on par with EMLP which only admits solutions with perfect symmetry. The results in Figure 3.2 (top) show that although the prior over models as described RPP-EMLP is broader than that of EMLP (as we can admit non-equivariant solutions) in the presence of perfectly equivariant data RPP-EMLP do not hinder performance, and we are able

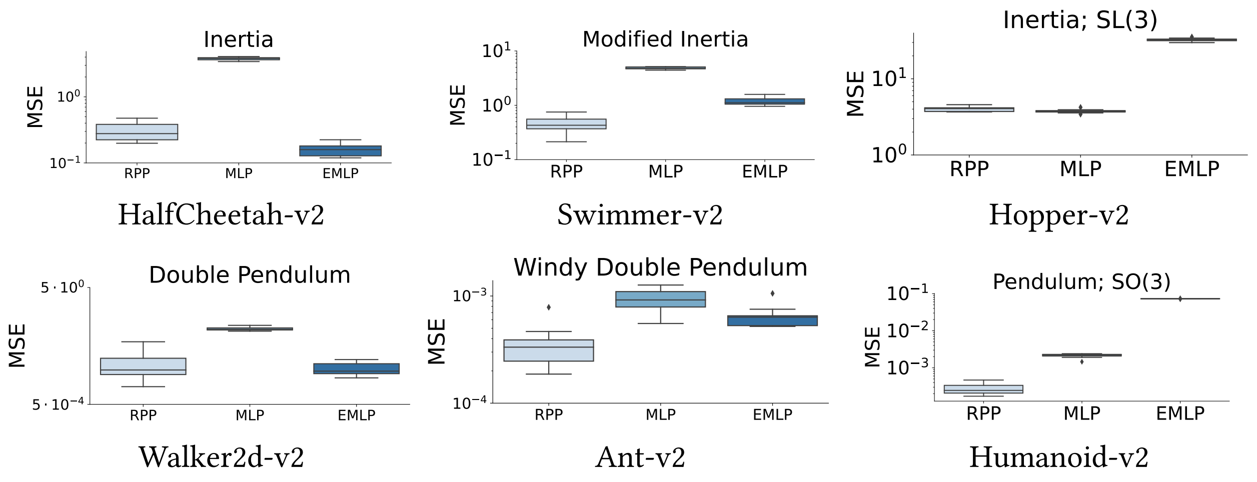


Figure 3.2: A comparison of test performance over 10 independent trials using RPP-EMLP and equivalent EMLP and MLP models on the inertia (**top**) and double pendulum (**bottom**) datasets in which we have three varying levels of symmetries. The boxes represent the interquartile range, and the whiskers the remainder of the distribution. **Left:** perfect symmetries in which EMLP and the equivariant components of RPP-EMLP exactly capture the symmetries in the data. **Center:** approximate symmetries in which the perfectly symmetric systems have been modified to include some non-equivariant components. **Right:** mis-specified symmetries in which the symmetric components of EMLP and RPP-EMLP do not reflect the symmetries present in the data.

to generalize nearly as well as the perfectly prescribed EMLP model.

Approximate symmetries To better showcase the ideas of Figure 3.1 we compare RPP-EMLPs to EMLPs and MLPs on the modified inertia and windy pendulum datasets. In these datasets we can think about the systems as primarily equivariant but containing non-equivariant contributions. As shown in 3.2 (bottom) these problems are best suited for RPP-EMLP as MLPs have no bias towards the approximately symmetry present in the data, and EMLPs are overly constrained in this setting.

Misspecified symmetries In contrast to working with perfect symmetries and showing that RPP-EMLPs are competitive with EMLPs, we also show that when symmetries are *mis*specified the bias towards equivariant solutions does not hinder the performance of RPP-EMLPs. For the inertia dataset we substitute the group equivariance in EMLP from O(3) to the overly large group

SL(3) consisting of all volume and orientation preserving linear transformations, not just the orthogonal ones. For the double pendulum dataset, we substitute O(2) symmetry acting on $\mathbb{R}^3$ with the larger SO(3) rotation group that contains it but is not a symmetry of the dataset.

By purposefully misspecifying the symmetry in these datasets we intentionally construct EMLP and RPP-EMLP models with incorrect inductive biases. In this setting EMLP is incapable of making accurate predictions as it has a hard constraint on an incorrect symmetry. Figure 3.2 shows that even in cases where the model is intentionally mis-specified that RPPs can overcome a poorly aligned inductive bias and recover solutions that perform as well as standard MLPs, even where EMLPs fail.

As the prior variances over the equivariant basis Q and the non-equivariant basis P describe our bias towards or away from equivariant solutions we investigate how the choice of prior variance relates to the level of symmetry present in a given dataset. In the windy pendulum dataset we have control over the level of wind and thus how far our system is from perfect equivariance.

3.5.2 Posterior Levels of Equivariance

RPPs describe a method for setting a prior over equivariance, and in the presence of new data we expect the posterior distribution over equivariance to change accordingly. Using samples from a deep ensemble to query points of high density in the posterior we estimate how the distribution over equivariance error progresses through training. Recalling that with an equivariant function f we have $\rho_2(g)f(x) = f(\rho_1(g)x)$, we compute equivariance error as

$$\text{EquivErr}(f, x) = \text{RelErr}(\rho_2(g)f(x), f(\rho_1(g)x)) \text{ where } \text{RelErr}(a, b) = \frac{\|a - b\|}{\|a\| + \|b\|}. \tag{3.3}$$

We train one deep ensemble on the inertia dataset which exhibits perfect symmetry, and another on the modified inertia dataset which has only partial symmetry, with each deep ensemble being comprised of 10 individual models using the same procedure as in Section 3.5.1. In Figure

?? (left) we see that throughout training the models trained on the modified inertia concentrate around solutions with substantially higher equivariance error than models trained on the dataset with the exact symmetry. This figure demonstrates one of the core desiderata of RPPs: that we are able to converge to solutions with an appropriate level of equivariance for the data.

3.5.3 RPPs and Convolutional Structure

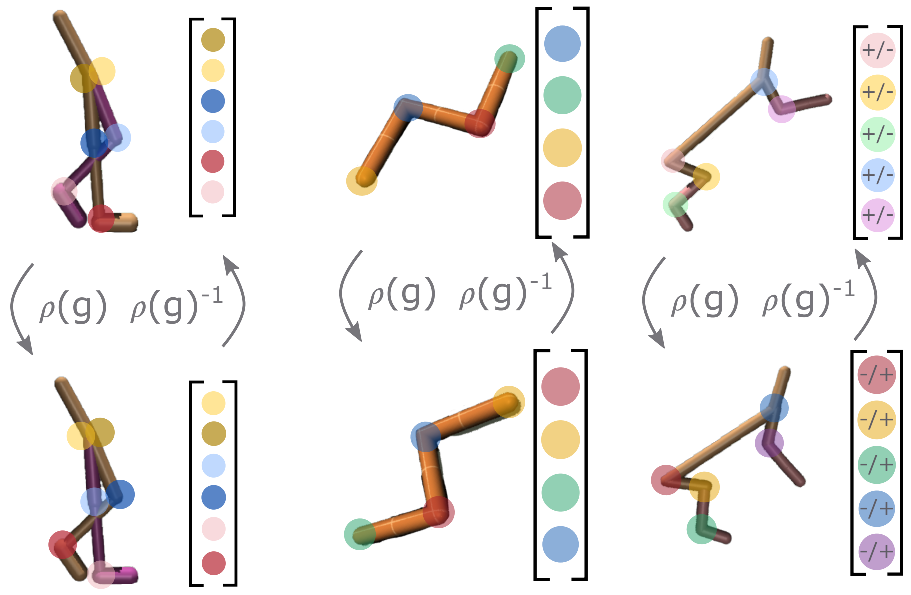
Using the RPP-Conv specified by the prior in Eqn 3.2 we apply the model to CIFAR-10 classification and UCI regression tasks where the inputs are reshaped to zero-padded two dimensional arrays and treated as images. Notably, the model is still an MLP and merely has larger prior variance in the convolutional subspace. As a result it can perform well on image datasets where the inductive bias is aligned, as well as on the UCI data despite not being an image dataset as shown in Table 3.1. While retaining the flexibility of an MLP, the RPP performs better than the locally connected MLPs trained with β-lasso in Neyshabur [2020] which get 14% error on CIFAR-10. The full details for the architectures and training procedure are given in Appendix A.2.3.

	CIFAR-10	Energy	Fertility	Pendulum	Wine
MLP	37.61 ± 0.56	0.39 ± 0.48	0.049 ± 0.0044	4.65 ± 0.50	0.66 ± 0.058
RPP	12.62 ± 0.34	0.73 ± 0.44	0.060 ± 0.0097	4.25 ± 0.50	0.69 ± 0.031
Conv	12.03 ± 0.46	1.34 ± 0.38	0.076 ± 0.0157	4.63 ± 0.36	0.79 ± 0.092

Table 3.1: Mean test classification error on CIFAR-10 and MSE on 4 UCI regression tasks, with one standard deviation errors taken over 10 trials. Similar to Figure 3.2, we find that whether the constrained convolutional structure is helpful (CIFAR) or not (UCI), RPP-Conv performs similarly to the model with the correct level of complexity.

3.6 Approximate Symmetries in Reinforcement Learning

Both model free and model based reinforcement learning present opportunities to take advantage of structure in the data for predictive power and data efficiency. On the one hand stands the



(a) Walker2d Left-Right (b) Swimmer Front-Back (c) HalfCheetah In-Out

Figure 3.3: Example illustrations of symmetries and representations from the Mujoco environments. **Left:** left-right symmetry in the *Walker2d* environment, **center:** front-back symmetry in the *Swimmer* environment, and **right:** In-out similarity in the *HalfCheetah* environment

use of model predictive control in the engineering community where finely specified dynamics models are constructed by engineers and only a small number of parameters are fit with system identification to determine mass, inertia, joint stiffness, etc. On the other side of things stands the hands off approach taken in the RL community, where general and unstructured neural networks are used for both transition models [Chua et al. 2018; Wang and Ba 2019; Janner et al. 2019] as well as policies and value functions [Haarnoja et al. 2018a]. The state and action spaces for these systems are highly complex with many diverse inputs like quaternions, joint angles, forces, torques that each transform in different ways under a symmetry transformation like a left-right reflection or a rotation. As a result, most RL methods treat these spaces a black box ignoring all of this structure, and as a result they tend to require tremendous amounts of training data, making it difficult to apply to real systems without the use of simulators.

We can make use of this information about what kinds of objects populate the state and action spaces to encode approximate symmetries of the RL environments. As shown in van der Pol et al. [2020b], exploiting symmetries in MDPs by using equivariant networks can yield substantial improvements in data efficiency. But symmetries are brittle, and minor effects like rewards for moving in one direction, gravity, or even perturbations like wind, a minor tilt angle in CartPole, or other environment imperfections can break otherwise perfectly good symmetries. As shown in Table 3.2, broadening the scope to approximate symmetries allows for leveraging a lot more structure in the data which we can exploit with RPP. While Walker2d, Swimmer, Ant, and Humanoid have exact left/right reflection symmetries, Hopper, HalfCheetah, and Swimmer have approximate front/back reflection symmetries. Ant and Humanoid have an even more diverse set, with the D_4 dihedral symmetry by reflecting and cyclicly permuting the legs of the ant, as well as continuous rotations of the Ant and Humanoid within the environment which can be broken by external forces or rewards. Identifying this structure in the data, we are able to use the generality of EMLP to construct an equivariant model for this data, and then turn it into a soft prior using RPP.

Symmetries	Walker2d	Hopper	HalfCheetah	Swimmer	Ant	Humanoid
Exact	$\mathbb{Z}_2$	✗	✗	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2$
Approximate	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2 \times \mathbb{Z}_2$	$D_4 \times O(2)$	$\mathbb{Z}_2 \times O(2)$
This work	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2 \times \mathbb{Z}_2$	$\mathbb{Z}_4$	$SO(2)$

Table 3.2: Exact and approximate symmetries of Mujoco locomotion environments of which we use the subgroups in the bottom row, see subsection A.2.4 for the detailed action and state representations.

3.6.1 Approximate Symmetries in Model Free Reinforcement Learning

We evaluate RPPs on the standard suite of Mujoco continuous control tasks in the context of model-free reinforcement learning. With the appropriately specified action and state representations detailed in subsection A.2.4, we construct RPP-EMLPs which we use as a drop-in replacement for both the policy and Q-function in the Soft Actor Critic (SAC) algorithm [Haarnoja et al. 2018a],

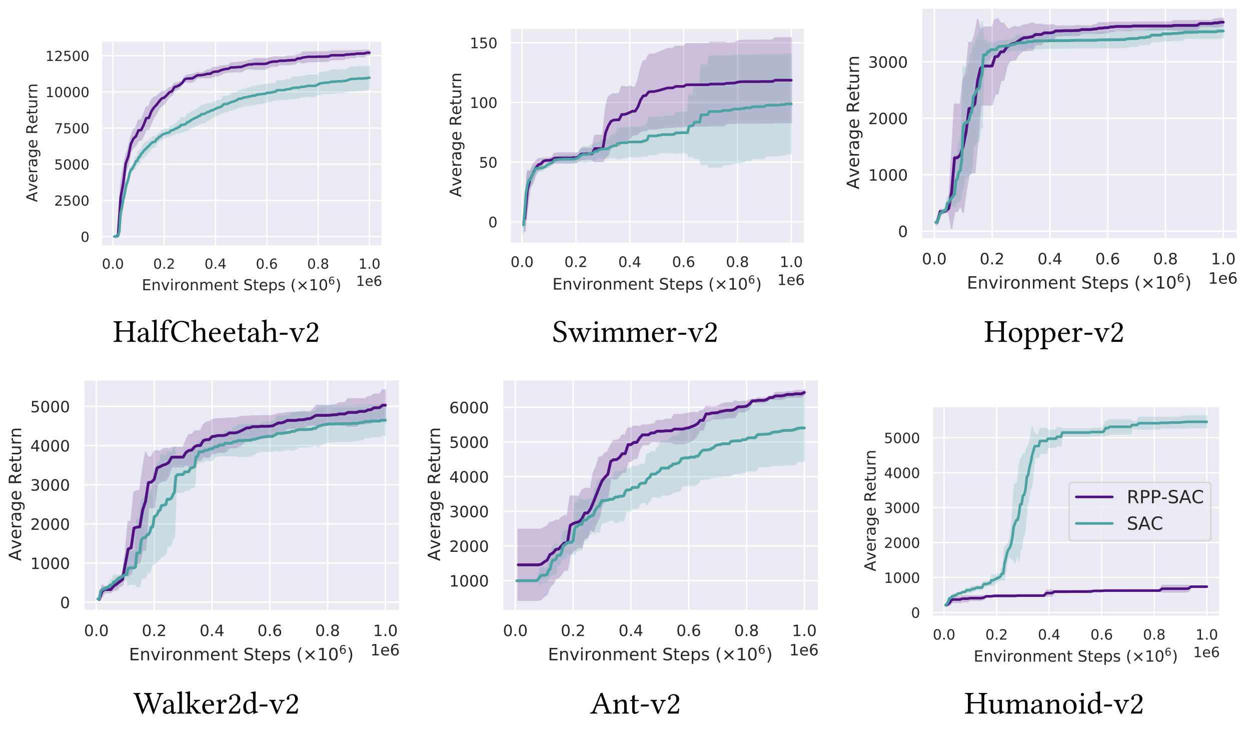


Figure 3.4: Average reward curve of RPP-SAC and SAC trained on Mujoco locomotion environments (max average reward attained at each step). Mean and one standard deviation taken over 4 trials shown in the shaded region. Incorporating approximate symmetries in the environments improves the efficiency of the model free RL agents.

using the same number of layers and channels. In contrast with van der Pol et al. [2020b] where equivariance is used just for policies, we find that using RPP-EMLP for the policy function alone is not very helpful with Actor Critic (see Figure 3.4). With the exception of the Humanoid-v2 environment where the RPP-EMLP destabilizes SAC, we find that incorporating the exact and approximate equivariance with RPP yields consistent improvements in the data efficiency of the RL agent as shown in Figure 3.4.

3.6.2 Better Transition Models for Model Based Reinforcement Learning

	Swimmer-v2		Hopper-v2		Ant-v2	
Rollout	MLP	**RPP**	MLP	**RPP**	**MLP**	RPP
10 Steps	0.51 ± 0.02	**0.40 ± 0.04**	1.1 ± 0.1	**0.9 ± 0.1**	**4.2 ± 0.1**	5.2 ± 0.3
30 Steps	1.6 ± 0.2	**1.26 ± 0.14**	3.8 ± 0.3	**3.1 ± 0.5**	**11.3 ± 0.2**	13.9 ± 0.7
100 Steps	3.9 ± 1.0	**2.75 ± 0.31**	9.8 ± 0.5	**7.0 ± 0.7**	**16.0 ± 0.3**	20.0 ± 1.1
Equiv Err	46%	19%	98%	32%	36%	31%

Table 3.3: Transition model rollout relative error in percent % averaged over 10, 30, and 100 step rollouts (geometric mean over trajectory). Errorbars are 1 standard deviation taken over 3 random seeds. Equivariance error is computed from as the geometric mean averaged over the 100 step rollout.

We also investigate whether the equivariance prior of RPP can improve the quality of the predictions for transition models in the context of model based RL. To evaluate this in a way decoupled from the complex interactions between policy, model, and value function in MBRL, we instead construct a static dataset of $50,000$ state transitions sampled uniformly from the replay buffer of a trained SAC agent. Since the trajectories in the replay buffer come from different times, they capture the varied dynamics MBRL transition models often encounter during training.

State of the art model based approaches on Mujoco tend to use an ensemble of small MLPs that predict the state transitions [Chua et al. 2018; Wang and Ba 2019; Janner et al. 2019; Amos et al. 2020], without exploiting any structure of the state space. We evaluate test rollout predictions via the relative error of the state over different length horizons for the RPP model against an MLP, the method of choice. As shown in Table 3.3, RPP transition models outperform MLPs on the Swimmer and Hopper environments, especially for long rollouts showing promise for use in MBRL. On these environments, RPP learns a smaller but non-negligible equivariance error that still enables it to fit the data.

3.7 Limitations

Using RPP-EMLP for the state and action spaces of the Mujoco environments required identifying the meaning of each of the components in terms of whether they are scalars, velocity vectors, joint angles, or orientation quaternions, and also which part of the robot they correspond to. This can be an error-prone process. While RPPs are fairly robust to such mistakes, the need to identify components makes using RPP more challenging than standard MLP. Additionally, due to the bilinear layers within EMLP, the Lipschitz constant of the network is unbounded which can lead to training instabilities when the inputs are not well normalized. We hypothesize these factors may contribute to the training instability we experienced using RPP-EMLP on Humanoid-v2.

3.8 Conclusion

In this work we have presented a method for converting restriction priors such as equivariance constraints into flexible models that have a bias towards structure but are not constrained by it. Given uncertainty about the nature of the data, RPPs are a safe choice. These RPP models are able to perform as well as the equivariant models when exact symmetries are present, and as well as unstructured MLPs when the specified symmetry is absent, and better than both for approximate symmetries. We have shown that encoding approximate symmetries can be a powerful technique for improving performance in a variety of settings, particularly for the messy and complex state and action spaces in reinforcement learning.

We hope that RPP enables designing more expressive priors for neural networks that capture the kinds of high level assumptions that machine learning researchers actually hold when developing models, rather than low level characteristics about the parameters that are hard to interpret. Building better techniques for enforcing high level properties helps lower the cost of incorporating prior knowledge, and better accommodate the complexities of data, even if they don't match our expectations.

4 | The Lie Derivative for Measuring Learned Equivariance

Equivariance guarantees that a model's predictions capture key symmetries in data. When an image is translated or rotated, an equivariant model's representation of that image will translate or rotate accordingly. The success of convolutional neural networks has historically been tied to translation equivariance directly encoded in their architecture. The rising success of vision transformers, which have no explicit architectural bias towards equivariance, challenges this narrative and suggests that augmentations and training data might also play a significant role in their performance. In order to better understand the role of equivariance in recent vision models, we introduce the Lie derivative, a method for measuring equivariance with strong mathematical foundations and minimal hyperparameters. Using the Lie derivative, we study the equivariance properties of hundreds of pretrained models, spanning CNNs, transformers, and Mixer architectures. The scale of our analysis allows us to separate the impact of architecture from other factors like model size or training method. Surprisingly, we find that many violations of equivariance can be linked to spatial aliasing in ubiquitous network layers, such as pointwise non-linearities, and that as models get larger and more accurate they tend to display more equivariance, regardless of architecture. For example, transformers can be more equivariant than convolutional neural networks after training.

This chapter is adapted from the paper "The Lie Derivative for Measuring Learned Equivariance", which will appear at ICLR 2023 and is joint work with Nate Gruver, Micah Goldblum, and Andrew Gordon Wilson.

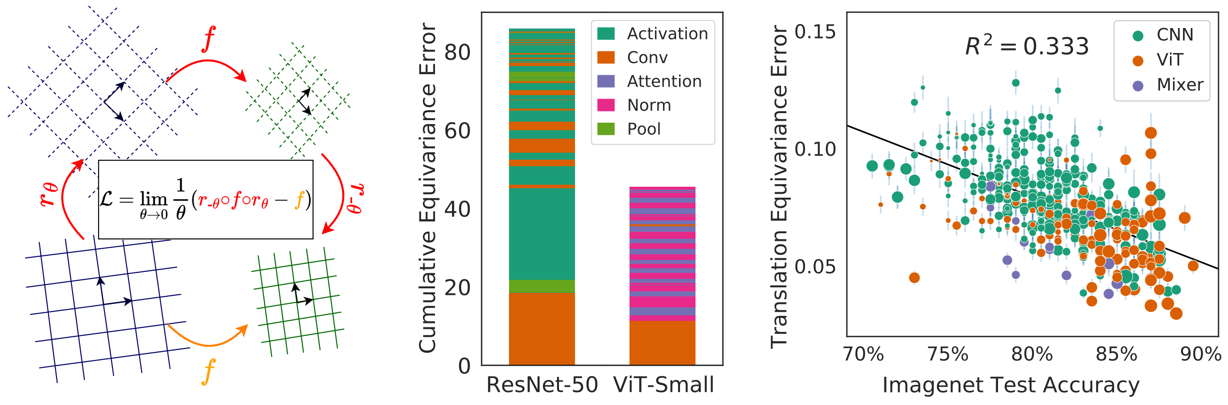


Figure 4.1: (Left): The Lie derivative measures the equivariance of a function under continuous transformations, here rotation. **(Center)**: Using the Lie derivative, we quantify how much each layer contributes to the equivariance error of a model. Our analysis highlights surprisingly large contributions from non-linearities, which affects both CNNs and ViT architectures. **(Right)**: Translation equivariance as measured by the Lie derivative correlates with generalization in classification models, across convolutional and non-convolutional architectures. Although CNNs are often noted for their intrinsic translation equivariance, ViT and Mixer models are often more translation equivariant than CNN models after training.

4.1 Introduction

Symmetries allow machine learning models to generalize properties of one data point to the properties of an entire class of data points. A model that captures translational symmetry, for example, will have the same output for an image and a version of the same image shifted a half pixel to the left or right. If a classification model produces dramatically different predictions as a result of translation by half a pixel or rotation by a few degrees it is likely misaligned with physical reality. Equivariance provides a formal notion of consistency under transformation. A function is equivariant if symmetries in the input space are preserved in the output space.

Baking equivariance into models through architecture design has led to breakthrough performance across many data modalities, including images [Cohen and Welling 2016a; Veeling et al. 2018], proteins [Jumper et al. 2021] and atom force fields [Batzner et al. 2022; Frey et al. 2022]. In computer vision, translation equivariance has historically been regarded as a particularly compelling property of convolutional neural networks (CNNs) [LeCun et al. 1995]. Imposing equivariance restricts the size of the hypothesis space, reducing the complexity of the learning problem and improving generalization [Goodfellow et al. 2016].

In most neural networks classifiers, however, true equivariance has been challenging to achieve, and many works have shown that model outputs can change dramatically for small changes in the input space [Azulay and Weiss 2018; Engstrom et al. 2018; Vasconcelos et al. 2021; Ribeiro and Schön 2021]. Several authors have significantly improved the equivariance properties of CNNs with architectural changes inspired by careful signal processing [Zhang 2019; Karras et al. 2021], but non-architectural mechanisms for encouraging equivariance, such as data augmentations, continue to be necessary for good generalization performance [Wightman et al. 2021].

The increased prominence of non-convolutional architectures, such as vision transformers (ViTs) and mixer models, simultaneously demonstrates that explicitly encoding architectural biases for equivariance is not necessary for good generalization in image classification, as ViT models perform on-par with or better than their convolutional counterparts with sufficient data and well-chosen augmentations [Dosovitskiy et al. 2020; Tolstikhin et al. 2021]. Given the success of large flexible architectures and data augmentations, it is unclear what clear practical advantages are provided by explicit architectural constraints over learning equivariances from the data and augmentations. Resolving these questions systemically requires a unified equivariance metric and large-scale evaluation.

In what follows, we introduce the Lie derivative as a tool for measuring the equivariance of neural networks under continuous transformations. The *local equivariance error* (LEE), constructed with the Lie derivative, makes it possible to compare equivariance across models and to analyze the contribution of each layer of a model to its overall equivariance. Using LEE, we conduct a large-scale analysis of hundreds of image classification models. The breadth of this study allows us to uncover a novel connection between equivariance and model generalization, and the surprising result that ViTs are often more equivariant than their convolutional counterparts after training. To explain this result, we use the layer-wise decomposition of LEE to demonstrate how common building block layers shared across ViTs and CNNs, such as pointwise non-linearities, frequently give rise to aliasing and violations of equivariance.

We make our code publicly available at https://github.com/ngruver/lie-deriv.

4.2 Background

Groups and equivariance Equivariance provides a formal notion of consistency under transformation. A function $f : V_1 \to V_2$ is equivariant to transformations from a symmetry group G if applying the symmetry to the input of f is the same as applying it to the output

$$\forall g \in G : \quad f(\rho_1(g)x) = \rho_2(g)f(x), \tag{4.1}$$

where $\rho(g)$ is the *representation* of the group element, which is a linear map $V \to V$.

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. Here $x \in V_1 = V_2$ is an image and the representation $\rho_1 = \rho_2$ expresses translations of the image. The translation *invariance* of certain neural networks is also an expression of the equivariance property, but where the output vector space V_2 has the trivial $\rho_2(g) = I$ representation, such that model outputs are unaffected by translations of the inputs. Equivariance is therefore a much richer framework, in which we can reason about representations at the input and the output of a function.

Continuous signals The inputs to classification models are discrete images sampled from a *continuous* reality. Although discrete representations are necessary for computers, the goal of classification models should be learning functions that generalize in the real world. It is therefore useful to consider an image as a function $h : \mathbb{R}^2 \to \mathbb{R}^3$ rather than a discrete set of pixel values and broaden the symmetry groups we might consider, such as translations of an image by vector $\in \mathbb{R}^2$, rather than an integer number of pixels.

Fourier analysis is a powerful tool for understanding the relationship between continuous signals and discrete samples by way of frequency decompositions. Any $M \times M$ image, $h()$, can be constructed from its frequency components, H_{nm}, using a $2d$ Fourier series, $h() = \frac{1}{2\pi} \sum_{n,m} H_{nm} e^{2\pi i \cdot [n,m]}$ where $\in [0,1]^2$ and $n, m \in [\text{-}M/2, \text{-}M/2+1, \ldots, M/2]$, the bandlimit defined by the image size.

Aliasing Aliasing occurs when sampling at a limited frequency f_s, for example the size of an image in pixels, causes high frequency content (above the Nyquist frequency $f_s/2$) to be converted into spurious low frequency content. Content with frequency n is observed as a lower frequency contribution at frequency

$$\mathrm{Alias}(n) = \left\{ \begin{array}{ll} n \bmod f_s & \text{if } (n \bmod f_s) < f_s/2 \\ (n \bmod f_s) - f_s & \text{if } (n \bmod f_s) > f_s/2 \end{array} \right\}. \tag{4.2}$$

If a discretely sampled signal such as an image is assumed to have no frequency content higher than f_s, then the continuous signal can be uniquely reconstructed using the Fourier series and have a consistent continuous representation. But if the signal contains higher frequency content which gets aliased down by the sampling, then there is an ambiguity and exact reconstruction is not possible.

Aliasing and equivariance Aliasing is critically important to our study because it breaks equivariance to continuous transformations like translation and rotation. When a continuous image is translated its Fourier components pick up a phase:

$$h() \mapsto h(-) \implies H_{nm} \mapsto H_{nm} e^{-2\pi i \cdot [n,m]}.$$

However, when an aliased signal is translated, the aliasing operation A introduces a scaling factor:

$$H_{nm} \mapsto H_{nm} e^{-2\pi i (b_0 \mathrm{Alias}(n) + b_1 \mathrm{Alias}(m))}$$

In other words, aliasing causes a translation *by the wrong amount*: the frequency component H_{nm} will effectively be translated by $[(\mathrm{Alias}(n)/n)b_0, (\mathrm{Alias}(m)/m)b_1]$ which may point in a different direction than , and potentially even the opposite direction. Applying shifts to an aliased image will yield the correct shifts for true frequencies less than the Nyquist but incorrect shifts for frequencies higher than the Nyquist. Other continuous transformations, like rotation, create similar asymmetries.

Many common operations in CNNs can lead to aliasing in subtle ways, breaking equivariance in turn. Zhang [2019], for example, demonstrates that downsampling layers causes CNNs to have inconsistent outputs for translated images. The underlying cause of the invariance is aliasing, which occurs when downsampling alters the high frequency content of the network activations. The $M \times M$ activations at a given layer of a convolutional network have spatial Nyquist frequencies $f_s = M/2$. Downsampling halves the size of the activations and corresponding Nyquist frequencies. The result is aliasing of all nonzero content with $n \in [M/4, M/2]$. To prevent this aliasing, Zhang [2019] uses a local low pass filter (Blur-Pool) to directly remove the problematic frequency regions from the spectrum.

While studying generative image models, Karras et al. [2021] unearth a similar phenomenon in the pointwise nonlinearities of CNNs. Imagine an image at a single frequency $h() = \sin(2\pi \cdot [n, m])$. Applying a nonlinear transformation to h creates new high frequencies in the Fourier series, as illustrated in Figure 4.2. These high frequencies may fall outside of the bandlimit, leading to aliasing. To counteract this effect, Karras

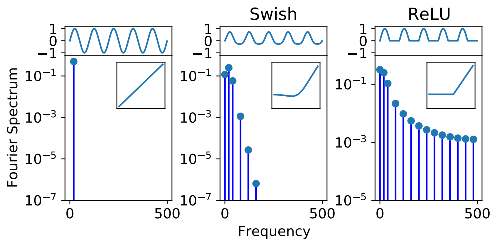


Figure 4.2: Non-linearities generate new high-frequency harmonics.

et al. [2021] opt for smoother non-linearities and perform upsampling before calculating the activations.

4.3 Related Work

While many papers propose architectural changes to improve the equivariance of CNNs [Zhang 2019; Karras et al. 2021; Weiler and Cesa 2019b], others focus purely on measuring and understanding how equivariance can emerge through learning from the training data [Lenc and Vedaldi 2015]. Olah et al. [2020], for example, studies learned equivariance in CNNs using model inversions techniques. While they uncover several fascinating properties, such as rotation equivariance that emerges naturally without architectural priors, their method is limited by requiring manual inspection of neuron activations. Most relevant to our work, Bouchacourt et al. [2021] measure equivariance in CNNs and ViTs by sampling group transformations. Parallel to our findings, they conclude that data and data augmentation play a larger role than architecture in the ultimate equivariance of a model. Because their study is limited to just a single ResNet and ViT architecture, however, they do not uncover the broader relationship between equivariance and generalization that we show here.

Many papers introduce consistency metrics based on sampling group transformations [Zhang 2019; Karras et al. 2021; Bouchacourt et al. 2021], but most come with significant drawbacks. When translations are sampled with an integer number of pixels [Zhang 2019; Bouchacourt et al. 2021], aliasing effects will be completely overlooked. As a remedy, [Karras et al. 2021] propose a subpixel translation equivariance metric (EQ-T_{frac}) that appropriately captures aliasing effects. While this metric is a major improvement, it requires many design decisions not required by LEE, which has relatively few hyperparameters and seamlessly breaks down equivariance across layers. Relative to these other approaches, LEE offers a unifying perspective, with significant theoretical and practical benefits.

4.4 Measuring local equivariance error with Lie derivatives

Lie Derivatives The Lie derivative gives a general way of evaluating the degree to which a function f violates a symmetry. To define the Lie derivative, we first consider how a symmetry group element can transform a function by rearranging Equation 4.1:

$$\rho_{21}(g)[f](x) = \rho_2(g)^{-1} f(\rho_1(g)x)\,.$$

The resulting linear operator, $\rho_{21}(g)[\cdot]$, acts on the vector space of functions, and $\rho_{21}(g)[f] = f$ if the function is equivariant. Every continuous symmetry group (Lie group), G, has a corresponding vector space (Lie algebra) $\mathfrak{g} = \mathrm{Span}(\{X_i\}_{i=1}^d)$, with basis elements X_i that can be interpreted as vector fields $\mathbb{R}^n \to \mathbb{R}^n$. For images, these vector fields encode infinitesimal transformations $\mathbb{R}^2 \to \mathbb{R}^2$ over the domain of continuous image signals $f : \mathbb{R}^2 \to \mathbb{R}^k$. One can represent group elements $g \in G$ (which lie in the connected component of the identity) as the *flow* along a particular vector field Φ_Y^t, where $Y = \sum_i a_i X_i$ is a linear combination of basis elements. The flow $\Phi_Y^t(x_0)$ of a point x_0 along a vector field Y by value t is defined as the solution to the ODE $\frac{dx}{dt} = Y(x)$ at time t with initial value x_0. The flow Φ_Y^t smoothly parameterizes the group elements by t so that the operator $\rho_{21}(\Phi_Y^t)[\cdot]$ connects changes in the space of group elements to changes in symmetry behavior of a function.

The Lie derivative of the function f is the derivative of the operator $\rho_{21}(g)$ at $g = \text{Identity} = \Phi_0$ along a particular vector field Y,

$$\mathcal{L}_Y(f) = \lim_{t \to 0} \frac{1}{t}(\rho_{21}(\Phi_Y^t)[f] - f) = \frac{d}{dt}\bigg|_{t=0} \rho_{21}(\Phi_Y^t)[f]. \tag{4.3}$$

Intuitively, the Lie derivative measures the sensitivity of a function to infinitesimal symmetry

```python
import torch.nn.functional as F
from torch.autograd.functional import jvp

def rotate(imgs, theta):
    """ Rotate images by angle theta and interpolate"""
    m = [[torch.cos(theta), torch.sin(theta), 0],
         [-torch.sin(theta), torch.cos(theta), 0]]
    m = torch.tensor(m)[None].expand(imgs.shape[0], -1, -1)
    return F.grid_sample(imgs, F.affine_grid(m, imgs.size(), True))

def rotation_lie_deriv(model,imgs):
    """ Lie deriv. of model w.r.t. rotation, can be scalar/image"""
    def rotated_model(theta):
        z = model(rotate(imgs,theta))
        img_like = (len(z.shape) == 4) # more complex for ViT/Mixer
        return rotate(z,-theta) if img_like else z
    return jvp(rotated_model, torch.zeros(1,requires_grad=True))[-1]

def e_lee(model,imgs):
    """ Expected equiv. error (E[|Lf|^2]/d_out) w.r.t. rotation"""
    return rotation_lie_deriv(model, imgs).pow(2).mean()
```

Figure 4.3: Lie derivatives can be computed using automatic differentiation. We show how a Lie derivative for continuous rotations can be implemented in PyTorch [Paszke et al. 2019]. The implementation in our experiments differs slightly, for computational efficiency and to pass second-order gradients through *grid_sample*.

transformations. This local definition of equivariance error is related to the typical global notion of equivariance error. As we derive in Appendix A.3.1.1, if $\forall i = 1, ..., d : \mathcal{L}_{X_i}(f) = 0$ (and the exponential map is surjective) then $\forall g \in G : f(\rho_1(g)x) = \rho_2(g)f(x)$ and for all x in the domain, and vice versa. In practice, the Lie derivative is only a proxy for strict global equivariance. We note global equivariance includes radical transformations like translation by 75% of an image, which is not necessarily desirable. In section 4.6 we show that our local formulation of the Lie derivative can capture the effects of many practically relevant transformations.

The Lie derivative of a function with multiple outputs will also have multiple outputs, so if we want to summarize the equivariance error with a single number, we can compute the norm of the Lie derivative scaled by the size of the output. Taking the average of the Lie derivative over the data distribution, we define Local Equivariance Error (LEE),

$$\mathrm{LEE}(f) = \mathbb{E}_{x\sim\mathcal{D}}\|\mathcal{L}_X f(x)\|^2/\dim(V_2). \tag{4.4}$$

We provide a Python implementation of the Lie derivative calculation for rotations in Figure 4.3 as an example. Mathematically, LEE also has an appealing connection to consistency regularization [Athiwaratkun et al. 2018], which we discuss in Appendix A.3.2.1.

Layerwise Decomposition of Lie Derivative Unlike alternative equivariance metrics, the Lie derivative decomposes naturally over the layers of a neural network, since it satisfies the chain rule. As we show in Appendix A.3.1.2, the Lie derivative of the composition of two functions $h : V_1 \to V_2$ and $f : V_2 \to V_3$ satisfies

$$\mathcal{L}_X(f\circ h)(x) = (\mathcal{L}_X f)(h(x)) + df|_{h(x)}(\mathcal{L}_X h)(x), \tag{4.5}$$

where $df|_{h(x)}$ is the Jacobian of f at $h(x)$ which we will abbreviate as df. Note that this decomposition captures the fact that intermediate layers of the network may transform in their own way:

$$f(h(x)) \mapsto \rho_{31}(g)[f\circ g](x) = \rho_3(g)^{-1}f(\rho_2(g)\rho_2(g)^{-1}h(\rho_1(g)x)) = \rho_{32}(g)[f]\circ\rho_{21}(g)[h](x)$$

and the Lie derivatives split up accordingly.

Applying this property to an entire model as a composition of layers $\mathrm{NN}(x) = f_{N:1}(x) := f_N(f_{N-1}(\ldots(f_1(x))))$, we can identify the contribution that each layer f_i makes to the equivariance error of the whole. Unrolling Equation 4.5, we have

$$\mathcal{L}_X(\mathrm{NN}) = \sum_{i=1}^{N} df_{N:i+1}\mathcal{L}_X f_i. \tag{4.6}$$

Intuitively, the equivariance error of a sequence of layers is determined by the sum of the equivariance error for each layer multiplied by the degree to which that error is attenuated or magnified by the other layers (as measured by the Jacobian). We evaluate the norm of each of the contributions $df_{N:i+1}\mathcal{L}_X f_i$ to the (vector) equivariance error $\mathcal{L}_X(\mathrm{NN})$ which we compute using autograd and stochastic trace estimation, as we describe in Appendix A.3.1.3. Importantly, the sum of norms may differ from the norm of the sum, but this analysis allows us to identify patterns across layers and pinpoint operations that contribute most to equivariance error.

4.5 Layerwise equivariance error

As described in section 5.2, subtle architectural details often prevent models from being perfectly equivariant. Aliasing can result from careless downsampling or from an activation function with a wide spectrum. In this section, we explore how the Lie derivative uncovers these types of effects automatically, across several popular architectures. We evaluate the equivariance of pretrained models on 100 images from the ImageNet [Deng et al. 2009] test set.

Using the layerwise analysis, we can dissect the sources of translation equivariance error in convolutional and non-convolutional networks as shown in Figure 4.4 (left) and (middle-left). For the Vision Transformer and Mixer models, we see that the initial conversion from image to patches produces a significant portion of the error, and the remaining error is split uniformly between the other nonlinear layers: LayerNorm, tokenwise MLP, and self-attention. The contribution from these nonlinear layers is seldom recognized and potentially counterintuitive, until we fully grasp the deep connection between equivariance and aliasing. In Figure 4.4 (middle-right), we show that this breakdown is strikingly similar for other image transformations like rotation, scaling, and hyperbolic rotations, providing evidence that the cause of equivariance error is not specific to translations but is instead a general culprit across a whole host of continuous transformations that can lead to aliasing.

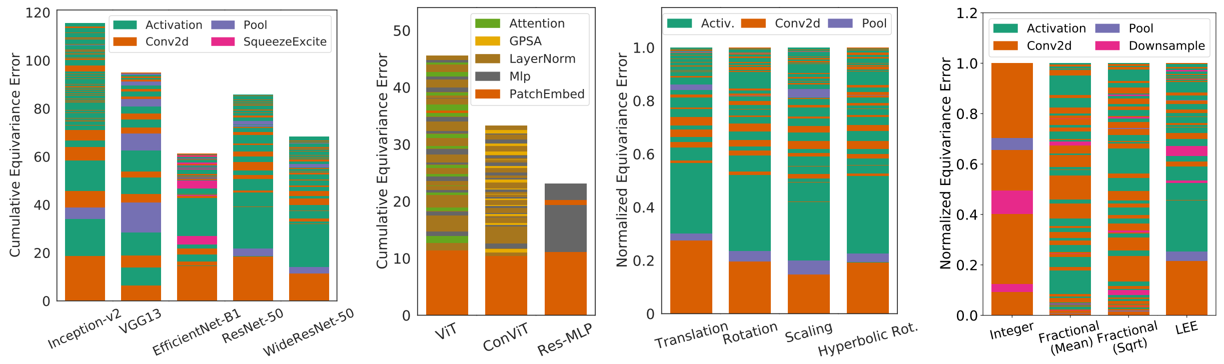


Figure 4.4: Contributions to equivariance shown cumulatively by layer, in the order the layers occur in the network. **Left**: Convolutional architectures. In all the CNNs, much of the equivariance error comes from downsampling and non-linearities. **Middle-Left**: Non-convolutional architectures. The initial patch embedding, a strided convolution, is the largest contributor for the ViTs and Mixers. The rest of the error is distributed uniformly across other nonlinear operations. **Middle-Right**: ResNet-50 across different transformations as a percentage. Despite being designed for translation equivariance, the fraction of equivariance error produced by each layer is almost identical for other affine transformations, suggesting that aliasing is the primary source of equivariance error. **Right**: Comparing LEE with alternative metrics for translation equivariance. Using integer translations misses key contributors to equivariance errors, such as activations, while using fractional translations can lead to radically different outcomes depending on choice of normalization (N or $\sqrt{N}$). LEE captures aliasing effects and has minimal design decisions.

We can make the relationship between aliasing and equivariance error precise by considering the aliasing operation Alias defined in Equation 4.2.

Theorem 4.1. *For translations along the vector $v = [v_x, v_y]$, the aliasing operation A introduces a translation equivariance error of*

$$\|\mathcal{L}_v(A)(h)\|^2 = (2\pi)^2 \sum_{n,m} H_{nm}^2 \big(v_x^2(\mathrm{Alias}(n) - n)^2 + v_y^2(\mathrm{Alias}(m) - m)^2\big),$$

where $h() = \frac{1}{2\pi}\sum_{n,m} H_{nm} e^{2\pi i\cdot[n,m]}$ is the Fourier series for the input image h.

We provide the proof in Appendix A.3.2.2. The connection between aliasing and LEE is important because aliasing is often challenging to identify despite being ubiquitous [Zhang 2019; Karras et al. 2021]. Aliasing in non-linear layers impacts all vision models and is thus a key factor in any fair comparison of equivariance.

As alternative equivariance metrics exist, it is natural to wonder whether they can also be used for layerwise analysis. In Figure 4.4 (right), we show how two equivariance metrics from Karras et al. [2021] compare with LEE, highlighting notable drawbacks. (1) Integer translation equivariance completely ignores aliasing effects, which are captured by both LEE and fractional translations. (2) Though fractional translation metrics (EQ-T$_{\text{frac}}$) correctly capture aliasing, comparing the equivariance of layers with different resolutions ($C \times H \times W$) requires an arbitrary choice of normalization. This choice can lead to radically different outcomes in the perceived contribution of each layer and is not required when using LEE, which decomposes across layers as described in section 4.4. We provide a detailed description of the baselines in Appendix A.3.3.1.

4.6 Trends in learned equivariance

Methodology We evaluate the Lie derivative of many popular classification models under transformations including $2d$ translation, rotation, and shearing. We define continuous transformations on images using bilinear interpolation with reflection padding. In total, we evaluate 410 classification models, a collection comprising popular CNNs, vision transformers, and MLP-based architectures [Wightman 2019]. Beyond diversity in architectures, there is also substantial diversity in model size, training recipe, and the amount of data used for training or pretraining. This collection of models therefore covers many of the relevant axes of variance one is likely to consider in designing a system for classification. We include an exhaustive list of models in the Appendix A.3.3.2.

Equivariance across architectures As shown in Figure 4.1 (right), the translation equivariance error (Lie derivative norm) is strongly correlated with the ultimate test accuracy that the model achieves. Surprisingly, despite convolutional architectures being motivated and designed for their translation equivariance, we find no significant difference in the equivariance achieved by convolutional architectures and the equivariance of their more flexible ViT and Mixer counterparts

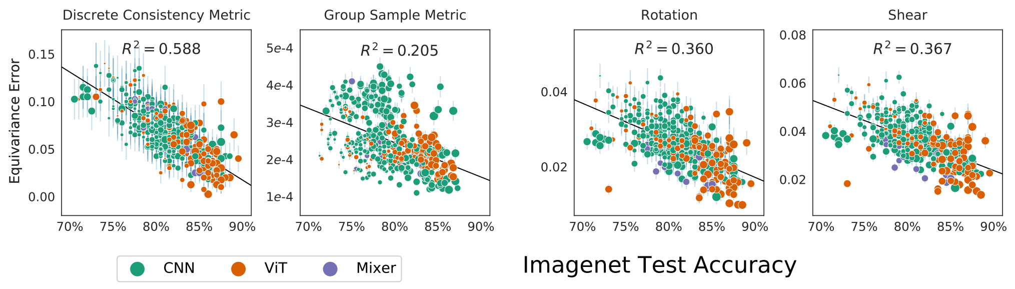


Figure 4.5: Equivariance metrics evaluated on the ImageNet test set. **Left:** Non-LEE equivariance metrics display similar trends to Figure 4.1, despite using larger, multi-pixel transformations. **Right:** Norm of rotation and shear Lie derivatives. Across all architectures, models with strong generalization become more equivariant to many common affine transformations. Marker size indicates model size. Error bars show one standard error over test set images used in the equivariance calculation.

when conditioning on test accuracy. This trend also extends to rotation and shearing transformations, which are common in data augmentation pipelines [Cubuk et al. 2020] (in Figure 4.5 (right)). Additional transformation results included in Appendix A.3.3.4.

For comparison, we also evaluate the same set of models using two alternative equivariance metrics: prediction consistency under discrete translation [Zhang 2019] and expected equivariance under group samples [Finzi et al. 2020; Hutchinson et al. 2021], which is similar in spirit to EQ-T_{frac} [Karras et al. 2021] (exact calculations in Appendix A.3.3.3). Crucially, these metrics are slightly less *local* than LEE, as they evaluate equivariance under transformations of up to 10 pixels at a time. The fact that we obtain similar trends highlights LEE's relevance beyond subpixel transformations.

Effects of Training and Scale In section 5.2 we described many architectural design choices that have been used to improve the equivariance of vision models, for example Zhang [2019]'s Blur-Pool low-pass filter. We now investigate how equivariance error can be reduced with non-architectural design decisions, such as increasing model size, dataset size, or training method. Surprisingly, we show that equivariance error can often be significantly reduced without any changes in architecture.

In Figure 4.6, we show slices of the data from Figure 4.1 along a shared axis for equivariance error. As a point of comparison, in Figure 4.6 (left), we show the impact of the Blur-Pool operation

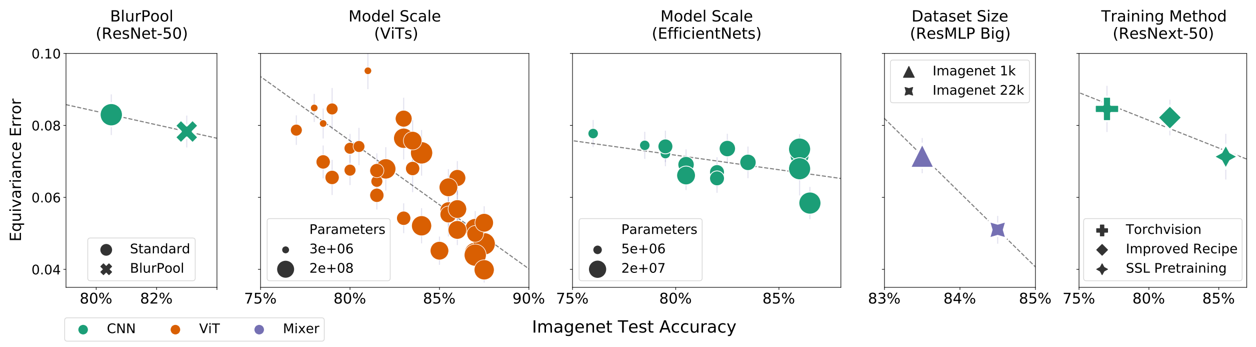


Figure 4.6: Case studies in decreasing translational equivariance error, numbered left-to-right. **1**: Blur-Pool [Zhang 2019], an architectural change to improve equivariance, decreases the equivariance error but by less than can be accomplished by improving the training recipe or increasing the scale of model or dataset. **2-3**: Increasing the number of parameters for a fixed model family (here ViTs [El-Nouby et al. 2021] and EfficientNets [Tan and Le 2019a]). **4**: Increasing the training dataset size for a ResMLP Big [Touvron et al. 2021a] model. **5**: Changing the training recipe for ResNeXt-50 [Xie et al. 2017] with improved augmentations [Wightman et al. 2021] or SSL pretraining [Yalniz et al. 2019]. Error bars show one standard error over images in the Lie derivative calculation.

discussed above on a ResNet-50 [Zhang 2019]. In the accompanying four plots, we show the effects of increasing model scale (for both ViTs and CNNs), increasing dataset size, and finally different training procedures. Although Zhang [2019]'s architectural adjustment does have a noticeable effect, factors such as dataset size, model scale, and use of modern training methods, have a much greater impact on learned equivariance.

As a prime example, in Figure 4.6 (right), we show a comparison of three training strategies for ResNeXt-50 – an architecture almost identical to ResNet-50. We use Wightman et al. [2021]'s pretrained model to illustrate the role of an improved training recipe and Mahajan et al. [2018b]'s semi-supervised model as an example of scaling training data. Notably, for a fixed architecture and model size, these changes lead to decreases in equivariance error on par with architectural interventions (BlurPool). This result is surprising when we consider that Wightman et al. [2021]'s improved training recipe benefits significantly from Mixup [Zhang et al. 2017] and CutMix [Yun et al. 2019], which have no obvious connection to equivariance. Similarly, Mahajan et al. [2018b]'s semi-supervised method has no explicit incentive for equivariance.

Equivariance out of distribution From our analysis above, large models appear to learn equivariances that rival architecture engineering in the classification setting. When learning equivariances through data augmentation, however, there is no guarantee that the equivariance will generalize to data that is far from the training distribution. Indeed, Engstrom et al. [2019] shows that carefully chosen translations or rotations can be as devastating to model performance as adversarial examples. We find that vision models do indeed have an *equivariance gap*: models are less equivariant on test data than train, and this gap grows for OOD inputs as shown in Figure 4.7. Notably, however, architectural biases do not have a strong effect on the equivariance gap, as both CNN and ViT models have comparable gaps for OOD inputs.

Why aren't CNNs more equivariant than ViTs? Given the deep historical connection between CNNs and equivariance, the results in Figure 4.5 and Figure 4.7 might appear counterintuitive. ViTs, CNNs, and Mixer have quite different inductive biases and therefore often learn very different representations of data [Raghu et al. 2021]. Despite their differences, however, all of these architectures are fundamentally constructed from similar building blocks–such as convolutions, normalization layers, and non-linear activations which can all contribute to aliasing and equivariance error. Given this shared foundation, vision models with high capacity and effective training recipes are more capable of fitting equivariances already present in augmented training data.

Model	Test Error (%)
G-CNN [36]	2.28
H-NET [216]	1.69
ORN [234]	1.54
TI-Pooling [119]	1.2
Finetuned MAE	1.14
RotEqNet [140]	1.09
E(2)-CNN [207]	0.68

Table 4.1: Our finetuned MAE is competitive with several architectures explicitly engineered to encode rotation invariance on RotMNIST, where rotation invariance is clearly crucial to generalization.

Learning rotation equivariance We finally consider the extent to which large-scale pretraining can match strong architectural priors in a case where equivariance is obviously desirable. We

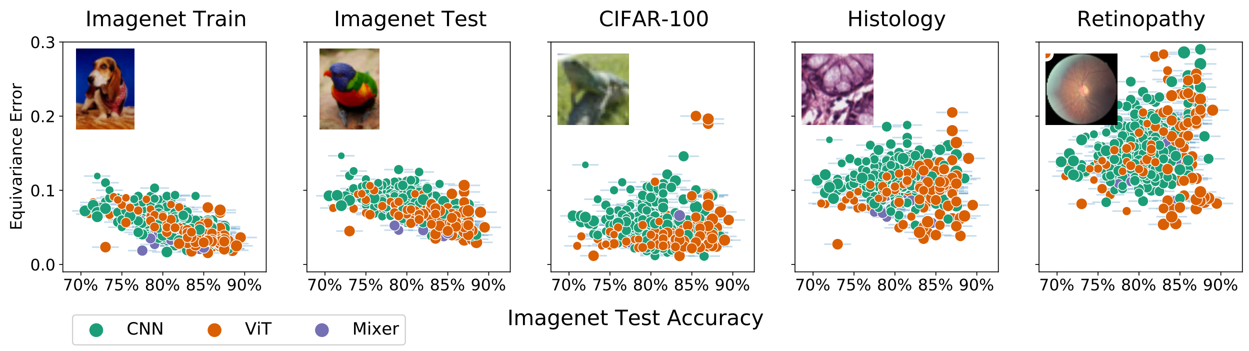


Figure 4.7: Models are less equivariant on test data and becoming decreasingly equivariant as the data moves away from the training manifold. As examples of data with similar distributions, we show equivariance error on the ImageNet train and test sets as well as CIFAR-100. As examples of out-of-distribution data, we use two medical datasets (which often use Imagenet pretraining), one for Histology [Kather et al. 2016] and one for Retinopathy [Kaggle and EyePacs 2015].

fine-tune a state-of-the-art vision transformer model pretrained with masked autoencoding [He et al. 2021] for 100 epochs on rotated MNIST [Weiler and Cesa 2019b] (details in Appendix A.3.3.5). This dataset, which contains MNIST digits rotated uniformly between -180 and 180 degrees, is a common benchmark for papers that design equivariant architectures. In Table 4.1 we show the test errors for many popular architectures with strict equivariance constrainets alongside the error for our finetuned model. Surprisingly, the finetuned model achieves competitive test accuracy, in this case a strong proxy for rotation invariance. Despite having relatively weak architectural biases, transformers are capable of learning and generalizing on well on symmetric data.

4.7 Conclusion

We introduced a new metric for measuring equivariance which enables a nuanced investigation of how architecture design and training procedures affect representations discovered by neural networks. Using this metric we are able to pinpoint equivariance violation to individual layers, finding that pointwise nonlinearities contribute substantially even in networks that have been designed for equivariance. We argue that aliasing is the primary mechanism for how equivariance to continuous transformations are broken, which we support theoretically and empirically. We use our measure to study equivariance learned from data and augmentations, showing model scale, data scale, or training recipe can have a greater effect on the ability to learn equivariances than architecture.

Many of these results are contrary to the conventional wisdom. For example, transformers can be more equivariant than convolutional neural networks after training, and can learn equivariances needed to match the performance of specially designed architectures on benchmarks like rotated MNIST, despite a lack of explicit architectural constraints. These results suggest we can be more judicious in deciding when explicit interventions for equivariance are required, especially in many real world problems where we only desire approximate and local equivariances.On the other hand, explicit constraints will continue to have immense value when exact equivariances and extrapolation are required — such as rotation invariance for molecules. Moreover, despite the ability to learn equivariances on training data, we find that there is an equivariance *gap* on test and OOD data which persists regardless of the model class. Thus other ways of combating aliasing outside of architectural interventions may be the path forward for improving the equivariance and invariance properties of models.

5 | Occam's razor and Understanding the Inductive Biases of Neural Networks

While there has been progress in developing non-vacuous generalization bounds for deep neural networks, these bounds tend to be uninformative about why deep learning works. In this chapter, we develop a compression approach based on quantizing neural network parameters in a linear subspace, profoundly improving on previous results to provide state-of-the-art generalization bounds on a variety of tasks, including transfer learning. We use these tight bounds to better understand the role of model size, equivariance, and the implicit biases of optimization, for generalization in deep learning. Notably, we find large models can be compressed to a much greater extent than previously known, encapsulating Occam's razor. We also argue for data-independent bounds in explaining generalization.

This chapter is adapted from the paper "PAC-Bayes Compression Bounds So Tight That They Can Explain Generalization", which originally appeared at NeurIPS 2022 and is joint work with Sanae Lotfi, Sanyam Kapoor, Andres Potap, Micah Goldblum, and Andrew Gordon Wilson.

5.1 Introduction

Despite many more parameters than the number of training datapoints, deep learning models generalize extremely well and can even fit random labels [Zhang et al. 2021]. These observations are not explained through classical statistical learning theory such as VC-dimension or Rademacher complexity which focus on uniform convergence over the hypothesis class [Nagarajan and Kolter 2019b]. The PAC-Bayes framework, by contrast, provides a convenient way of constructing generalization bounds where the generalization gap depends on the deep learning model found by training rather than the hypothesis set as a whole. Using this framework, many different potential explanations have been proposed drawing on properties of a deep learning model that are induced by the training dataset, such as low spectral norm [Neyshabur et al. 2018], noise stability [Arora et al. 2018b], flat minima [Hochreiter and Schmidhuber 1997], derandomization [Negrea et al. 2020], robustness, and compression [Arora et al. 2018b; Zhou et al. 2019].

In this work, we show that neural networks, when paired with structured training datasets, are substantially more compressible than previously known. Constructing tighter generalization bounds than have been previously achieved, we show that this compression *alone* is sufficient to explain many generalization properties of neural networks.

In particular:

1. We develop a new approach for training compressed neural networks that adapt the compressed size to the difficulty of the problem. We train in a random linear subspace of the parameters [Li et al. 2018] and perform learned quantization. Consequently, we achieve extremely low compressed sizes for neural networks at a given accuracy level, which is essential for our tight bounds. (See Section 5.4).

2. Using a prior encoding Occam's razor and our compression scheme, we construct the best generalization bounds to date on image datasets, both with data-dependent and data-

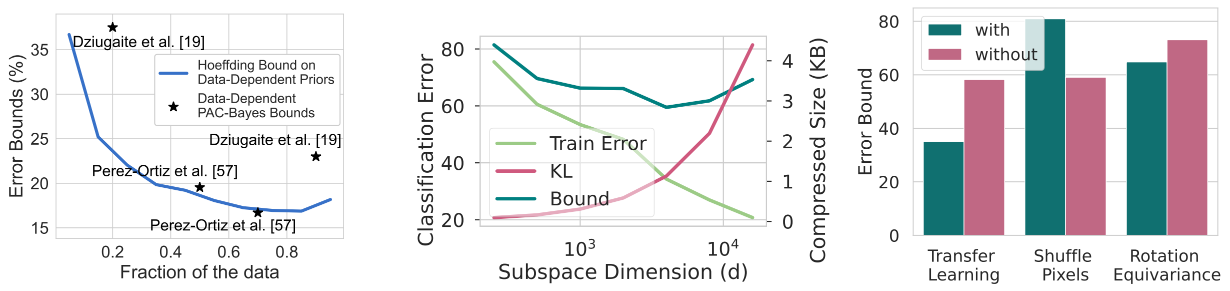
independent priors. We also show how transfer learning improves compression and thus our generalization bounds, explaining the practical performance benefits of pre-training. (See Section 5.5).

3. PAC-Bayes bounds only constrain the adaptation of the prior to the posterior. For bounds constructed with data-dependent priors, we show that the prior alone achieves performance comparable to the generalization bound. Therefore we argue that bounds constructed from data-independent priors are more informative for understanding generalization. (See Section 5.5.2).

4. Through the lens of compressibility, we are able to help explain why deep learning models generalize on structured datasets like CIFAR-10, but not when structure is broken such as by shuffling the pixels or shuffling the labels. Similarly, we describe the benefits of equivariant models, e.g. why CNNs outperform MLPs. Finally, we investigate double descent and whether the implicit regularization of SGD is necessary for generalization. (See Section 5.6).

We emphasize that while we achieve state-of-the-art results in both data-dependent bounds and data-independent bounds through our newly developed compression approach, our goal is to leverage these tighter bounds to understand generalization in neural networks. Among others, Figure 5.1 highlights some of our observations regarding (a) data-dependent bounds, (b) how our method trades-off between data fit and model compression in relation to generalization, and (c) the explanation of several deep learning phenomena through model compressibility using our bounds.

All code to reproduce results is available here.



(a) Pitfalls of data-dependent bounds (b) Adaptive compression (c) Deep learning phenomena

Figure 5.1: The power of data-independent subspace compression bounds for explaining deep learning phenomena. Bounds for CIFAR-10 except (c)-rotation, which is rotMNIST. **(a)** We show that the simple Hoeffding bound computed *only* on the data-dependent prior and evaluated on the remainder of the training data (essentially measuring validation loss) achieves error bounds that are competitive or even better than data-dependent bounds obtained by previous works, showing that data-dependent PAC-Bayes bounds do not explain generalization any further than the prior alone. Instead, data-independent bounds are more informative for understanding generalization (see Section 5.5.2). **(b)** Training error, the KL term (compressed model size measured in KB), and our PAC-Bayes bound as the subspace dimension is varied. For a fixed network, our method provides an adaptive compression scheme that trades off compressed size with training error, finding the optimal bound for a given model and dataset. **(c)** We compute our data-independent bounds for model trained *with* and *without*: transfer learning, shuffling the pixels, and the rotation-equivariance property. Our bounds identify the positive impact of transfer learning, how breaking structure in the data by shuffling pixels hurts the model, and that rotationally equivariant models improve generalization on rotated data. Each of these interventions impact the compressibility of the models. See Section 5.6 for more details.

5.2 Related Work

Optimizing the PAC-Bayes Bound. Dziugaite and Roy [2017] obtained the first non-vacuous generalization bounds for deep stochastic neural networks on binary MNIST. The authors constructed a relaxation of the Langford and Seeger [2001] bound and optimized it to find a posterior distribution that covers a large volume of low-loss solutions around a local minimum obtained using SGD. Rivasplata et al. [2019] further extended the idea by developing novel relaxations of PAC-Bayes bounds based on Blundell et al. [2015].

Model Compression and PAC-Bayes Bounds. Noting the robustness of neural networks to small perturbations [Hinton and Van Camp 1993; Hochreiter and Schmidhuber 1997; Langford and Caruana 2001; Langford 2002; Keskar et al. 2017; Neyshabur et al. 2018; Chaudhari et al.

2017], Arora et al. [2018b] developed a compression-based approach that uses noise stability. Additionally, they used the ability to reconstruct weight matrices with random projections to study generalization of neural networks. Subsequently, Zhou et al. [2019] developed a PAC-Bayes bound that uses the representation of a compressed model in bits, and added noise stability through the use of Gaussian posteriors and Gaussian mixture priors. Furthermore, they achieved even smaller model representations through pruning and quantization [Han et al. 2016; Cheng et al. 2018]. Our compression framing is similar to Zhou et al. [2019] but with key improvements. First, we train in a lower dimensional subspace using intrinsic dimensionality [Li et al. 2018] and FiLM subspaces [Perez et al. 2018] which proves to be more effective and adaptable than pruning. Second, we develop a more aggressive quantization scheme with variable length code and quantization aware training. Finally, we exploit the increased compression provided by transfer learning and data-dependent priors.

Data-Dependent Priors. Dziugaite et al. [2021] demonstrated that for linear PAC-Bayes bounds such as Thiemann et al. [2017], a tighter bound can be achieved by choosing the prior distribution to be data-dependent, i.e., the prior is trained to concentrate around low loss regions on held-out data. More precisely, the authors show that the optimal data-dependent prior is the conditional expectation of the posterior given a subset of the training data. They approximate this data-dependent prior by solving a variational problem over Gaussian distributions. They evaluate the bounds for SGD-trained networks on data-dependent priors obtaining tight bounds on MNIST, Fashion MNIST, and CIFAR-10. In a similar vein, Pérez-Ortiz et al. [2021] combine data-dependent priors [Dziugaite et al. 2021] with the PAC-Bayes with Backprop (PBB) [Rivasplata et al. 2019] to obtain *state-of-the-art* PAC-Bayes non-vacuous bounds for MNIST and CIFAR-10 using data-dependent priors.

Downstream Transferability. Ding et al. [2022] investigate different correlates of generalization derived from PAC-Bayes bounds to predict the transferability of various upstream models; however, because of this different aim they do not actually compute the full bounds.

Table 5.1: Non-vacuous PAC-Bayes bounds obtained on popular image classification datasets in deep learning. ⋆ indicates bounds obtained using data-dependent priors (Section 5.5.2). ✗ indicates that either the method does not support multi-class problems or that it is completely reliant on data-dependent priors and therefore cannot result in data-independent bounds. Additionally, we add Binary MNIST for reference to a benchmark used in earlier works.

	Non-vacuous PAC-Bayes bounds (%)					
Reference	Binary MNIST	MNIST	FMNIST	CIFAR-10	CIFAR-100	ImageNet
Dziugaite and Roy [2017]	16.1	✗	✗	✗	✗	✗
Rivasplata et al. [2019]	2.2	✗	✗	✗	✗	✗
Zhou et al. [2019]		46	91.6	100	100	96.5
Dziugaite et al. [2021]		11⋆	38⋆	23⋆	✗	✗
Pérez-Ortiz et al. [2021]		21.7/1.5⋆	49.1	90.0/16.7⋆	100	✗
Our bounds		**11.6/1.4⋆**	**32.8/10.1⋆**	**58.2/16.6⋆**	**94.6/44.4⋆**	**93.5/40.9⋆**

Our focus is to achieve better bounds in order to better understand generalization in deep neural networks. For example, we investigate the effects of transfer learning, equivariance, and stochastic training on the bounds, and argue for the importance of data-independent bounds in explaining generalization. We summarize improvements of our bounds relative to prior results in Table 5.1.

5.3 A Primer on PAC-Bayes Bounds

PAC-Bayes bounds are fundamentally an expression of Occam's razor: simpler descriptions of the data generalize better. As an illustration, consider the classical generalization bound on a finite hypothesis class. Let $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h\left(x_i\right), y_i\right)$ be the empirical risk of a hypothesis $h \in \mathcal{H}$, with $|\mathcal{H}| < \infty$. Let ℓ be the 0-1 loss, and let $R(h) = \mathbb{E}[\hat{R}(h)]$ denote the population risk. With probability at least $1-\delta$, the population risk of hypothesis h using n data samples satisfies

$$R(h) \leqslant \hat{R}(h) + \sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{2n}}. \tag{5.1}$$

In other words, the population risk is bounded by the empirical risk and a complexity term $\log |\mathcal{H}|$ which counts the number of bits needed to specify any hypothesis $h \in \mathcal{H}$.

But what if we don't believe that each hypothesis is equally likely? If we consider a prior distribution over the hypothesis class that concentrates around likely hypotheses, then we can construct a variable length code that uses fewer bits to specify those hypotheses. Note that if P is a prior distribution over $\mathcal{H}$, then any given hypothesis h will take $\log_2 \frac{1}{P(h)}$ bits to represent when using an optimal compression code for P. This prior may result in a smaller complexity term as long as the hypotheses that are consistent with the data are also likely under the prior, regardless of the size of the hypothesis class.

Moreover, the number of bits required can be reduced from $\log_2 \frac{1}{P(h)}$ to $\mathbb{KL}(Q \parallel P)$ by considering a distribution of "good" solutions Q. If we don't care which element of Q we arrive at, we can gain some bits *back* from this insensitivity (which could be used to code a separate message). The average number of bits to code a sample from Q using the prior P is the cross entropy $\mathbb{H}(Q, P)$ and we get $\mathbb{H}(Q)$ bits back from being agnostic about which sample $h \sim Q$ to use, yielding the KL-divergence between Q and P: $\mathbb{H}(Q, P) - \mathbb{H}(Q) = \mathbb{KL}(Q \parallel P)$.

With these improvements on the finite hypothesis bound — replacing $\log |\mathcal{H}|$ with $\mathbb{KL}(Q \parallel P)$, and sampling a hypothesis $h \in \mathcal{H}$ — we arrive (with minor bookkeeping) at the PAC-Bayes bound introduced in McAllester [1999]. This last bound states that with probability at least $1 - \delta$,

$$\mathop{\mathbb{E}}_{h \sim Q} [R(h)] \leqslant \mathop{\mathbb{E}}_{h \sim Q} [\hat{R}(h)] + \sqrt{\frac{\mathbb{KL}(Q \parallel P) + \log(n/\delta) + 2}{2n - 1}}. \tag{5.2}$$

Many refinements of (5.2) have been developed [Langford and Seeger 2001; Maurer 2004; Catoni 2007; Thiemann et al. 2017] but retain the same character. That is, the lower the ratio of the KL-divergence to the number of data points n, the lower the gap between empirical and expected risk. In this work, we use the tighter Catoni [2007] variant of the PAC-Bayes bound (see Appendix A.4.10 for details).

Universal Prior. Leveraging Occam's razor, we can define a prior that explicitly penalizes the minimum compressed length of the hypothesis, also known as the universal prior [Solomonoff 1964]: $P(h) = 2^{-K(h)}/Z$, where K is the *prefix* Kolmogorov complexity [Hutter et al. 2008] of h (the length of the shortest program that produces h and also delimits itself), and $Z \leqslant 1$.[1] Using a point mass posterior on a single hypothesis h^*, we get the following upper-bound

$$\mathbb{KL}\left(\mathbf{1}_{[h=h^*]} \parallel P(h)\right) = \log \tfrac{1}{P(h^*)} \leqslant K(h^*) \log 2 \leqslant l(h^*) \log 2 + 2 \log l(h^*),$$

where $l(h)$ is the length of a given program that reproduces h not including the delimiter. For convenience, we can condition on using the same method for compression and decompression for all elements of the prior. Lastly, we can improve the tightness of the previous PAC-Bayes bound by reducing the compressed length $l(h^*)$ of the hypothesis h^* that we found during training.

Model Compression. *Model compression* aims to find nearly equivalent models that can be expressed in fewer bits either for deploying them in mobile devices or for improving their inference time on specialized hardware [Cheng et al. 2017, 2018]. For computing PAC-Bayes generalization bounds, however, we only care about the model size. Therefore, we can employ compression methods which may otherwise be unfavorable in practice due to worse computational requirements. *Pruning* and *quantization* are among the most widely used methods for model compression. In this work, we rely on quantization (Section 5.4.2) to achieving tighter generalization bounds.

5.4 Tighter Generalization Bounds via Adaptive Subspace Compression

Training a neural network involves taking many gradient steps in a high-dimensional space $\mathbb{R}^D$. Although D may be large, the loss landscape has been found to be simpler than typically believed

[1]The universal prior similar to the discrete hypothesis prior from Zhou et al. [2019] but setting $m(h) = 2^{-2\log_2 l(h)}$ rather than the flat $m(h) = 2^{-72}$.

[Dauphin et al. 2014; Goodfellow and Vinyals 2015; Garipov et al. 2018]. Analogous to the notion of intrinsic dimensionality more generally, Li et al. [2018] searched for the lowest dimensional subspace in which the network can be trained and still fit the training data. The weights of a neural network $\theta \in \mathbb{R}^D$ are parametrized in terms of an initialization θ_0 and a projection $w \in \mathbb{R}^d$ to a lower dimensional subspace through a fixed matrix $P \in \mathbb{R}^{D\times d}$,

$$\theta = \theta_0 + Pw. \tag{5.3}$$

To facilitate favorable conditioning during optimization, P is chosen to be approximately orthonormal $P^\top P \approx I_{d\times d}$. For scalability, Li et al. [2018] use random normal matrices of the form $P \sim \mathcal{N}(0,1)^{D\times d}/\sqrt{D}$, and their sparse approximations [Li et al. 2006; Le et al. 2013].

In its original form, intrinsic dimensionality (ID) is only used as a scientific tool to measure the complexity of the learning task. Unlike methods like pruning, intrinsic dimension [Li et al. 2018] scales with complexity of the task — more complex tasks require a larger intrinsic dimension. Subsequently, we find that ID combined with quantization can serve as an effective model compression method. We note that ideas similar to the intrinsic dimensionality of a model have been explored in the context of model compression for estimating bounds [Arora et al. 2018b]. For our work, the ability to find the intrinsic dimension $d \ll D$ has profound implications for the compressibility of the models, and therefore our ability to construct generalization bounds. As demonstrated by Zhou et al. [2019], the compressibility of a neural network has a direct connection to generalization and allows us to compute non-vacuous PAC-Bayes bounds for transfer learning.

Resting upon ID, our key building blocks to achieve tight generalization bounds are composed of (i) a new scalable method to train an intrinsic dimensionality neural network parameterized by (5.3) (Section 5.4.1), and (ii) a new approach to simultaneously train both the quantized neural network weights and the quantization levels for maximum compression (Section 5.4.2). Our complete method is summarized in Algorithm 2.

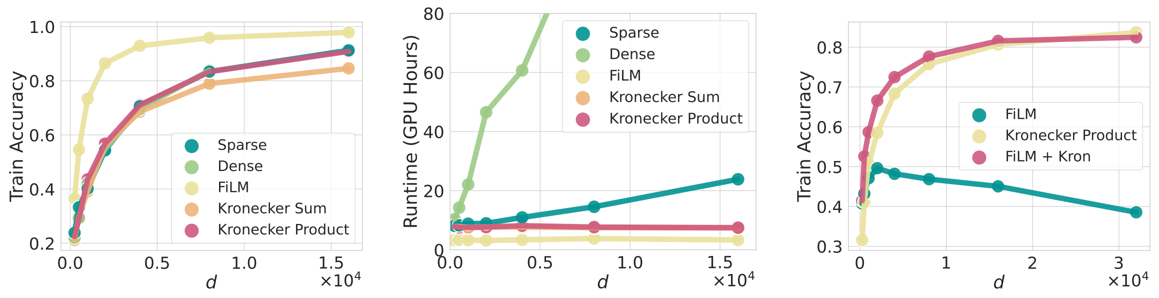


Figure 5.2: Effective and scalable projection operators. (Left) Different projection operators P (Section 5.4.1) used for transfer learning from Imagenet to CIFAR-10 on a ResNet-34 across different subspace dimensions d. Kronecker product, Sparse, and Dense perform almost identically **(Center)** Kronecker product runs with substantially reduced the runtime cost compared to the Sparse or Fastfood matrices used by Li et al. [2018]. **(Right)** Training from scratch on CIFAR-10. The FiLM projector alone is unable to fit the data when training from scratch, and instead a sum of FiLM and Kronecker Product projectors perform the best.

5.4.1 Finding Better Random Subspaces

To further improve upon the scalability and effectiveness of the projections P used by Li et al. [2018], we introduce three novel projector constructions.

Kronecker Sum Projector. Using the Kronecker product $\otimes$, we construct the matrix $P_\oplus = (\mathbf{1} \otimes R_1 + R_2 \otimes \mathbf{1})/\sqrt{2D}$ where $R_1, R_2 \sim \mathcal{N}(0,1)^{\sqrt{D}\times d}$ and $\mathbf{1}$ is the vector of all ones in $\mathbb{R}^{\sqrt{D}}$. Noting that $R_1 \perp\!\!\!\perp R_2$ and that the entries are standard normal, $P_\oplus^\top P_\oplus = I_{d\times d} + \mathcal{O}(1/\sqrt{D})$.

Kronecker Product Projector. Alternatively, we form the matrix $P_\otimes = Q_1 \otimes Q_2/\sqrt{D}$ with the smaller $Q_1, Q_2 \sim \mathcal{N}(0,1)^{\sqrt{D}\times\sqrt{d}}$, and again this matrix is approximately orthogonal: $P_\otimes^\top P_\otimes = (Q_1^\top Q_1/\sqrt{D}) \otimes (Q_2^\top Q_2/\sqrt{D}) = I \otimes I + O(D^{-1/4}) = I_{d\times d} + O(D^{-1/4})$. [2]

The matrix vector multiply $w \mapsto Pw$ for both of the above projectors can be performed in time $O(d\sqrt{D})$ and $O(\sqrt{dD})$ respectively, rather than the $O(dD)$ that is required by the dense random matrix. Figure 5.2 demonstrates the runtime speedup and training performance improvement in comparison to the methods using by Li et al. [2018]. Notably, the Kronecker-structured projections

[2]As neither D nor d is typically a perfect square, we concatenate a dense random matrix to pad out the difference between D, d, and a perfect square. As $(\sqrt{D}+1)^2 = D + 2\sqrt{D} + 1$, we have that the size of this padding is at most $\sqrt{D} \times \sqrt{d}$, so it does not increase the asymptotic cost of performing the matrix vector multiplies.

retain the fidelity of the dense random matrix while being orders of magnitude faster than the alternative operators when scaling to larger values of d. In other words, the Kronecker-structured projectors are as good as the dense projectors for generating random linear subspaces of a given size, but are much scalable.

FiLM projector. BatchNorm parameters have an disproportionate effect on the downstream task performance relative to their size. This observation has been used in Featurewise independent Linear Modulation (FiLM) [Perez et al. 2018; Dumoulin et al. 2018] for efficient control of neural networks in many different settings. Several authors have explored performing fine-tuning for transfer learning solely on these parameters and the final linear layer [Kanavati and Tsuneki 2021]. Drawing on these observations, we construct a projection matrix P_{FiLM} where only columns corresponding with BatchNorm or head parameters are non-zero and sampled from $\mathcal{N}(0,1)^d/\sqrt{D}$, which we also show in Figure 5.2. While the FiLM projector is highly effective for transfer learning (shown in Figure 5.2 left), the performance saturates quickly when training from scratch. For this reason, when training from scratch we employ the sum $P_{\text{FiLM}+\otimes} = (P_{\text{FiLM}} + P_{\otimes})/\sqrt{2}$, which outperforms the two projectors individually as shown in Figure 5.2 right.

5.4.2 Quantization Scheme and Training

Through quantization, the average number of bits used per parameter can be substantially reduced. When optimizing purely for model size rather than efficiency on specialized hardware, we can choose non-linearly spaced quantization levels which are learned, and use variable length coding schemes as shown in Han et al. [2016]. Additionally, the straight through estimator has been central to learning weights in binary neural networks [Hubara et al. 2016]. We combine these ideas to simultaneously optimize the quantized weights and the quantization levels for maximum compression.

Given the full precision weights $w = [w_1, \dots, w_d] \in \mathbb{R}^d$ and a vector $c = [c_1, \dots c_L] \in \mathbb{R}^L$ of L quantization levels, we construct the quantized vector $\hat{w} = [\hat{w}_1, \dots, \hat{w}_d]$ such that $\hat{w}_i = c_{q(i)}$

where $q(i) =_k |w_i - c_k|$. The quantization levels c are learned alongside w, where the gradients are defined using the straight through estimator [Bengio et al. 2013; Yin et al. 2019]:

$$\frac{\partial \hat{w}_i}{\partial w_j} = \delta_{ij} \quad \text{and} \quad \frac{\partial \hat{w}_i}{\partial c_k} = 1_{[q(i)=k]} \tag{5.4}$$

We initialize c with uniform spacing between the minimum and maximum values in parameter vector w or k-means [Choi et al. 2017]. To further compress the network, we use a variable length code in the form of arithmetic coding [MacKay and Mac Kay 2003], which takes advantage of the fact that certain quantization levels are more likely than others. Given probabilities p_k (empirical fractions) for cluster c_k, arithmetic coding of w takes at most $\lceil d \times \mathbb{H}(p) \rceil + 2$ bits, where $\mathbb{H}(p)$ is the entropy $\mathbb{H}(p) = -\sum_k p_k \log_2 p_k$. For a small number of quantization levels, arithmetic coding yields better compression than Huffman coding.

In summary, we use $\lceil d \times \mathbb{H}(p) \rceil + 2$ bits for coding the quantized weights $\hat{w}$, $16L$ bits for the codebook c (represented in half precision), and additional $L \times \lceil \log_2 d \rceil$ bits for representing the probabilities p_k, arriving at $l(w) \leqslant \lceil d \times \mathbb{H}(p) \rceil + L \times (16 + \lceil \log_2 d \rceil) + 2$. As we show in Appendix A.4.2, we optimize over the subspace dimension d and the number of quantization levels L and any other hyperparameters, by including them in the compressed description of our model, contributing only a few extra bits.

5.4.3 Transfer Learning

For transfer learning, we replace θ_0 with a learned initialization $\theta_{\mathcal{D}}$ that is found using the pretraining task and data $\mathcal{D}$. With the ID compression, the universal prior $P(h \mid \theta_{\mathcal{D}}) \propto 2^{-K(h|\theta_{\mathcal{D}})}$ will place higher likelihood on solutions θ that are close to the pre-training solution $\theta_{\mathcal{D}}$.

Algorithm 2 Compute PAC-Bayes Bound.

```
0: **Inputs:** Neural network $f_\theta$, Training dataset $\{x_i, y_i\}_{i=1}^n$, Clusters $L$, Intrinsic dimension $d$,
   Confidence $1 - \delta$, and Prior distribution $P$.
0: function COMPUTE_BOUND($f_\theta, L, d, (x_i, y_i)_{i=1}^n, \delta, P$)
0:     $w \leftarrow$ TRAIN_ID($f_\theta$, $d$, $(x_i, y_i)_{i=1}^n$) {(Section 5.4.1)}
0:     $\hat{w} \leftarrow$ TRAIN_QUANTIZE($w$, $L$, $(x_i, y_i)_{i=1}^n$)
0:     Compute quantized train error $\hat{R}(\hat{w})$.
0:     $\mathbb{KL}(Q, P) \leftarrow$ GET_KL($\hat{w}, P$) {(Section 5.3)}
0:     return GET_CATONI_BOUND($\hat{R}(\hat{w})$, $\mathbb{KL}(Q, P)$, $\delta$, $n$) {(Section 5.3)}
0: end function
0: function TRAIN_QUANTIZE($w, L, (x_i, y_i)_{i=1}^n$) {(Section 5.4.2)}
0:     Initialize $c \leftarrow$ GET_CLUSTERS($w, L$) for $i = 1$ to quant_epochs do
0:
   $c \leftarrow c - \rho \nabla_c \mathcal{L}(w, c)$ and $w \leftarrow w - \rho \nabla_w \mathcal{L}(w, c)$
0:
0:     return $\hat{w}$
0: end function
0: function GET_KL(($\hat{w}, P$))
0:     $c$, count $\leftarrow$ GET_UNIQUE_VALS_COUNTS($\hat{w}$)
0:     message_size $\leftarrow$ DO_ARITHMETIC_ENCODING($\hat{w}$, $c$, count)
0:     message_size $\leftarrow$ message_size + hyperparam_search {(Appendix A.4.2)}
0:     return message_size + $2 \times \log$ (message_size)
0: end function=0
```

5.5 Empirical Non-Vacuous Bounds

Combining the training in structured random subspaces with our choice of learned quantization, we produce extremely compressed but high performing models. Using the universal prior, we bound the generalization error of these models and optimize over the degree of compression via the subspace dimension and other hyperparameters as summarized in Algorithm 2. We additionally describe hyperparameters, architecture specifications for each experiment, and other experimental details in Appendix A.4.5. In the following subsections, we apply our method to generate strong generalization bounds in the data-independent, data-dependent, and transfer learning settings.

Table 5.2: Our PAC-Bayesian subspace compression bounds compared to *state-of-the-art* (SOTA) bounds. All results are with 95% confidence, i.e. $\delta = .05$. The sign † refers to data-independent SOTA numbers that we computed using [Pérez-Ortiz et al. 2021], which we run on the additional datasets.

Dataset	Data-independent priors		Data-dependent priors	
	Err. Bound (%)	SOTA (%)	Err. Bound (%)	SOTA (%)
MNIST	**11.6**	21.7 [Pérez-Ortiz et al. 2021]	**1.4**	1.5 [Pérez-Ortiz et al. 2021]
+ SVHN Transfer	**9.0**	$16.1^{\dagger}$		
FashionMNIST	**32.8**	$46.5^{\dagger}$	**10.1**	38 [Dziugaite et al. 2021]
+ CIFAR-10 Transfer	**28.2**	$30.1^{\dagger}$		
CIFAR-10	**58.2**	$89.9^{\dagger}$	**16.6**	16.7 [Pérez-Ortiz et al. 2021]
+ ImageNet Transfer	**35.1**	$54.2^{\dagger}$		
CIFAR-100	**94.6**	$100^{\dagger}$	**44.4**	–
+ ImageNet Transfer	**81.3**	$98.1^{\dagger}$		
ImageNet	**93.5**	96.5 [Zhou et al. 2019]	**40.9**	–

5.5.1 Non-Vacuous PAC-Bayes Bounds

We present our bounds for the *data-independent* prior in Table 5.2. We derive the first non-vacuous bounds on FashionMNIST, CIFAR-10, and CIFAR-100 without data-dependent priors. These results have particular significance, as we argue in Section 5.5.2 that using data-dependent priors are not explanatory about the learning process. In particular, we improve over the compression bound results obtained by Zhou et al. [2019] on MNIST from 46% to 11.55% and on ImageNet from 96.5% to 94.1%. In terms of compression, we dramatically improve the rates as we reduce the compressed size for the best MNIST bound by 94% bringing it down from 6.23 KB to 0.38KB with LeNet5 and, on ImageNet, by 87% bringing it down from 358 KB to 46.3 KB with MobileViT. Since we perform transfer learning with an ImageNet-trained checkpoint, we omit transfer learning experiments on the ImageNet (downstream) dataset. The tightness of our SOTA subspace compression bounds allows us to improve the understanding of several deep learning phenomena as discussed in Section 5.6. See **??** A.4.5.1 for model architectures and Appendix A.4.1 for additional results.

5.5.2 Data-Dependent PAC-Bayes Bounds

So far, we demonstrated the strength of our bounds on *data-independent* priors, where we considerably improve on the state-of-the-art. However, a number of recent papers have considered data-dependent priors as a way of achieving tighter bounds [Pérez-Ortiz et al. 2021; Dziugaite et al. 2021]. In this setup, the training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ is partitioned into two parts, $\mathcal{D}_a$ and $\mathcal{D}_b$, with length $n - m$ and m. The first dataset is used to construct a data-dependent prior $P(h \mid \mathcal{D}_a)$, and then the bound is formed over the remaining part of the process: the adaptation of the prior $P(h \mid \mathcal{D}_a)$ to the posterior $Q(h)$ using the data $\mathcal{D}_b$. The empirical risk is computed over $\mathcal{D}_b$ only. Intuitively, using dataset $\mathcal{D}_a$ it is possible to construct a much tighter prior over the possible neural network solutions. In our setting, similar to transfer learning, we use the prior $P_{\mathcal{D}_a}(\theta) = 2^{-K(\theta|\theta_{\mathcal{D}_a})}/Z$ where for compression we use $\theta = \theta_{\mathcal{D}_a} + Pw$, and $\theta_{\mathcal{D}_a}$ is the solution found by training the model (without random projections) on the data $\mathcal{D}_a$ rather than initializing randomly. With these data-dependent priors, we achieve the best bounds in Table 5.2.

However, our adaptive approach exposes a significant downside of data-dependent priors. To the extent that PAC-Bayes bounds can be used for explanation, data-dependent bounds only provide insights into the procedure used to adapt the prior $P_{\mathcal{D}_a}(\theta)$ to the posterior Q using $\mathcal{D}_b$: any learning that is done in finding $P_{\mathcal{D}_a}(\theta)$ is not constrained or explained by the bound. Given the ability to adapt the size of the KL to the difficulty of the problem, it is possible to squeeze all of the learning into $P_{\mathcal{D}_a}(\theta)$ and none in this adaption to Q. This phenomenon happens as the $\mathbb{KL} \to 0$, which we find happens empirically (or very nearly so) across splits of the data, and especially when $n - m$ is large. Setting $Q(\theta) = \mathbf{1}_{[\theta = \theta_{\mathcal{D}_a}]}$, the KL has only the contribution from the optimization over d: $\mathbb{KL}(Q||P_{\mathcal{D}_a}) \leqslant \log D$. We find that the bound is nothing more than a variant of the simple Hoeffding bound where $\mathcal{D}_b$ is the validation set $R\left(\theta_{\mathcal{D}_a}\right) \leqslant \hat{R}_{\mathcal{D}_b}\left(\theta_{\mathcal{D}_a}\right) + \sqrt{\frac{\log(Dm/\delta)+2}{2m-1}}$.

We can see this phenomenon in Figure 5.1(a) where we compare existing data-dependent bounds to the simple Hoeffding bound applied directly to the data-dependent prior which was trained on only a small fraction of the data. We can consider the Hoeffding bound as the simplest data-dependent bound without any fine-tuning so that the *prior*, a single pre-trained checkpoint, is directly evaluated on held-out validation data with no KL-divergence term. If another data-dependent bound cannot achieve significantly stronger guarantees than the prior Hoeffding bound, then it only explains that neural networks generalize because the priors already have low validation error which is no explanation for generalization at all. Indeed, we see in Figure 5.1 that the strength of existing data-dependent bounds relies almost entirely on the a priori properties of the data-dependent prior rather than constraining the actual learning process through compressibility. Similarly, from a minimum description length (MDL) perspective, data-independent bounds can be used to provide a lossless compression of the training data, whereas data-dependent bounds cannot (see Appendix A.4.8).

We also note that with data-dependent priors, optimization over the subspace dimension selects very low dimensionality, even if the data does not have low intrinsic dimension. Because most of the data fitting is moved into fitting the prior, the bound selects a low complexity solution with respect to the prior without hurting data fit by choosing a low subspace dimensionality (Appendix A.4.4).

By contrast, data-independent bounds explain generalization for the entirety of the learning process. Similarly, our transfer learning bounds meaningfully constrain what happens in the fine-tuning on the downstream task, but they do not constrain the prior determined from the upstream task.

5.5.3 Non-Vacuous PAC-Bayes Bounds for Transfer Learning

By directly interpreting PAC-Bayes bounds through the lens of compression, we immediately see the benefits of using an upstream dataset for transfer learning. Transfer learning allows us to constrain the prior $P(\theta \mid \theta_{\mathcal{D}_a})$ around parameters consistent with the upstream dataset $\mathcal{D}_a$, reducing the KL-divergence between the prior and the posterior and leading to even tighter bounds as we show in Table 5.2. Our tighter data-independent transfer learning bounds provide a theoretical certification that transfer learning can improve generalization. Our PAC-Bayes transfer learning approach also indicates that transfer learning can boost generalization whenever codings optimized on a pre-training task are more efficient for encoding a downstream posterior than an a priori guess made before seeing data. By contrast, downstream tasks which greatly differ from the upstream task may only be consistent with models that are not compressible under the learned prior, a scenario that describes negative transfer. See Appendix A.4.3 for more details.

5.6 Understanding Generalization through PAC-Bayes Bounds

The classical viewpoint of uniform convergence focuses on properties of the hypothesis class as a whole, such as its size. In contrast, PAC-Bayes shows that the ability to generalize is not merely a result of the hypothesis class but also a result of the particular dataset and the characteristics of the individual functions in the hypothesis class. After all, many elements of our hypothesis class are not compressible, yet in order to guarantee generalization, we choose the ones that are. Real datasets actually contain a tremendous amount of structure, or else we could not learn from them as famously argued by Hume [1978] and No Free Lunch theorems [Wolpert and Macready 1997; Giraud-Carrier and Provost 2005]. This high degree of structure in real-world datasets is reflected in the compressibility of the functions (i.e. neural networks) we find in our hypothesis class which fit them.

In this section, we examine exactly how dataset structure manifests in compressible models by applying our generalization bounds, and we see what happens when this structure is broken, for example by shuffling pixels or fitting random labels. Corrupting the dataset degrades both compressibility and generalization.

MLPs vs CNNs. It is well known that convolutional neural networks (CNNs) generalize much better than standard multilayer perceptrons (MLPs) with alternating fully-connected layers and activation functions on image classification problems, even when controlling for the number of parameters. In our generalization bounds, this is reflected in the improved compressibility of CNNs when compared to MLPs. In Figure 5.3 (left), we see how on CIFAR-10 a we are able to

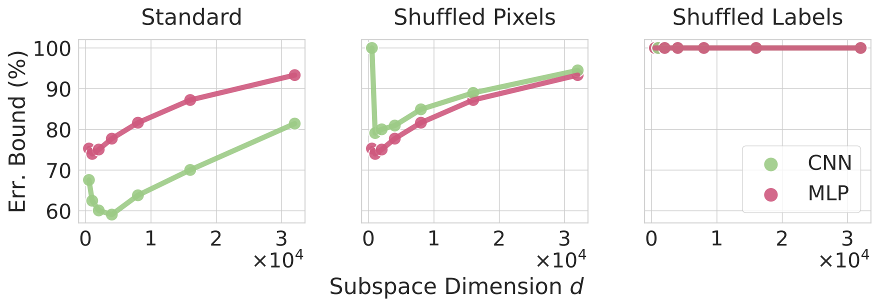


Figure 5.3: Breaking structure in the data and the model. Our PAC-Bayes bound computed using various subspace dimensions for a fixed size CNN and MLP, both with 500k parameters. We train on **(left)** CIFAR-10, **(center)** CIFAR-10 with shuffled pixels, **(right)** CIFAR-10 with shuffled labels. Structure in the dataset induces structure in the model. As structure is removed from the dataset, models which fit the data become much less compressible, hence generalize worse.

find a lower description length for a CNN than an MLP with the same number of parameters. See **??** A.4.5.4 for experimental details.

Shuffled Pixels. However, when the image structure is broken by shuffling the pixels, we find that CNNs are no better at generalizing than MLPs. For this dataset, CNNs become substantially less compressible and hence our bounds show them generalizing worse than MLPs, see Figure 5.3 (center). MLPs do not suffer when this structure is broken since they never used it in the first place.

Shuffled Labels. When the structure of the dataset is entirely broken by shuffling the labels, the compressibility of the models (both for CNNs and MLPs) which fit the random labels is lost. Regardless of the subspace dimension used, our generalization bounds are all at 100% error as shown in Figure 5.3 (right). It is not possible to fit the training data using low subspace dimensions, and when using a large enough dimension to fit the data, the compressed size of the model is larger than the training data and hence the generalization bounds are vacuous.

Equivariance. Designing models which are *equivariant* to certain symmetry transformations has been a guiding principle for the development of data-efficient neural networks in many domains [Cohen and Welling 2016a; Cohen et al. 2018b; Thomas et al. 2018; Weiler and Cesa 2019a; Finzi et al. 2020; Jumper et al. 2021]. While intuitively it is clear that respecting dataset symmetries severely improves generalization, relatively little has been proven for neural networks [Lyle et al. 2020; Elesedy and Zaidi 2021; Zhu et al. 2021; Bietti et al. 2021; Elesedy 2022]. We compress and evaluate rotationally equivariant (C_8) and non-equivariant Wide ResNets [Weiler and Cesa 2019a; Zagoruyko and Komodakis 2016] trained on MNIST and a rotated version of MNIST. As shown in Figure A.5, the rotationally equivariant models are more compressible and provably generalize better than their non-equivariant counterparts when paired with a dataset that also has the rotational symmetry. See Appendix A.4.6 for further details.

Is Stochasticity Necessary for Generalization? It is widely hypothesized that the implicit biases of SGD help to find solutions which generalize better. For example, Arora et al. [2018a] argue that there is no regularizer that replicates the benefits of gradient noise. Wu et al. [2020], Smith et al. [2020], and Li et al. [2021] advocate that gradient noise is necessary to achieve state-of-the-art performance. In comparison, recent work by Geiping et al. [2022] shows that full-batch gradient descent can match state-of-the-art performance, and Izmailov et al. [2021] shows that full-batch Hamiltonian Monte Carlo sampling generalizes significantly better than mini-batch MCMC and stochastic optimization.

We train ResNet-18 and LeNet5 models on CIFAR-10 and MNIST, respectively, using full-batch and SGD with different intrinsic dimensionalities. We provide the training details in Appendix A.4.7. For MNIST with LeNet5, the best generalization bounds that we obtain are 11.55% and 11.20% using stochastic gradient descent (SGD) and full-batch training respectively. The best generalization bounds that we obtain for CIFAR-10 with ResNet-18 are 74.68% and 75.3% using SGD and full-batch training respectively. We also extend this analysis to SVHN to MNIST transfer learning with LeNet5 and obtain PAC-Bayes bounds of 9.0% and 8.7% using SGD and full-batch training respectively. These close theoretical guarantees on the generalization error for both SGD and full-batch training suggest that while the implicit biases of SGD may be helpful, they are not at all necessary for understanding why neural networks generalize well (see Appendix A.4.7).

Double Descent. Our bounds are also tight enough to predict the double descent phenomenon noted in Nakkiran et al. [2020]. See Appendix A.4.9 for a depiction of these experiments and a discussion of their significance.

5.7 Discussion

In this work, we constructed a new method for compressing deep learning models that is highly adaptive to the model and to the training dataset. Following Occam's prior, which considers shorter compressed length models to be more likely, we construct state-of-the-art generalization bounds across a variety of settings. Through our compression bounds, we show how generalization relates to the structure in the dataset and in the model, and we are able to explain aspects of neural network generalization for natural image datasets, shuffled pixels, shuffled labels, equivariant models, and stochastic training.

Limitations. Despite the power of our compression scheme and the ability of our bounds to faithfully describe the generalization properties of many modeling decisions and phenomena, we are scratching the surface of explaining generalization. Our compression bounds prefer models with a smaller number of parameters as shown in Appendix A.4.8, instead of larger models which actually tend to generalize better. While we achieved better model compression than previous works, it is unlikely that we are close to theoretical limits. Maybe through nonlinear parameter compression schemes we might find that larger deep learning models are more compressible than smaller models. Moreover, it is unclear how to relate the bounds of the compressed models to their uncompressed counterparts, perhaps leveraging ideas from Nagarajan and Kolter [2019a] and others investigating this question. Additionally, while our bounds show that the compressibility of our models implies generalization, we make no claims about the reverse direction. However, we believe that model compression and Occam's razor have yet untapped explanatory power in deep learning.

6 | Conclusion

In conclusion, this thesis has advanced the understanding and application of mathematical inductive biases in the context of deep learning models, by developing novel methods for constructing equivariant models, and investigating the role of equivariance and other inductive biases in learning and generalization. The four chapters of this thesis present a comprehensive exploration of the interplay between equivariance, model construction, priors, and generalization, opening new avenues for research and practical applications in the field of machine learning.

Chapter 1 introduced EMLP, a general construction for equivariant linear layers that encompasses a wide variety of matrix groups and representations. This innovative approach has demonstrated consistently improved generalization in problems involving symmetry, such as Lorentz invariant particle scattering and dynamical systems. While there exists a tradeoff between generality and specialized implementation, EMLP shows great promise in reducing experimentation costs and fostering the development of new methods for using equivariant models on diverse domains. Future research may focus on overcoming the limitations in training speed and model size, allowing for even more versatile applications of this approach.

Chapter 2 presented RPP, a method for converting restriction priors, such as equivariance constraints, into flexible models that are biased towards structure but not constrained by it. RPP models have shown superior performance in various settings, particularly in reinforcement learning where state and action spaces are often messy, with only approximate symmetries. This work encourages the development of more expressive priors for neural networks, which can better capture high-level assumptions and accommodate the complexities of data, even when they do not match our expectations. The results from this chapter suggest that future research should focus on designing better techniques for enforcing high-level properties and incorporating prior knowledge more effectively.

Chapter 3 introduced a new metric for measuring equivariance, enabling a nuanced investigation of how architectural design and training procedures affect the representations discovered by neural networks. The findings of this chapter challenge conventional wisdom, highlighting that factors such as model scale, data scale, and training recipe can play a more significant role in learning equivariances than architecture. These results have important implications for the design and evaluation of neural networks, highlighting the impact of aliasing in symmetry breaking, the existence of an equivariance gap on out of distribution data, and highlighting elements outside of model construction that can help ameleriorate these factors through learning. Future research should explore alternative ways of combating aliasing to improve equivariance and smoothness properties in such models.

Finally, Chapter 4 constructed a new method for compressing deep learning models, proving state-of-the-art generalization bounds that adapt to both the model and the training dataset. This work has illuminated the relationships between generalization, dataset structure, and model structure, explaining aspects of neural network generalization across various settings through the lens of the Occam's razor prior. With this approach we are able to prove generalization bounds showing the benefits of equivariance when the data and model structure are aligned. The results of this chapter underline the ways in which the inductive biases of the model, the latent structure in the data, and the induced generalization properties interrelate.

In summary, this thesis has made significant strides in broadening the scope and automation of equivariant model construction, uncovering the role of inductive biases in learning and generalization, and developing new machine learning models for scientific applications. By building on the foundations established in this work, future research can further exploit the power of mathematical inductive biases, enabling the development of more intelligent, efficient, and effective deep learning models that reflect our prior knowledge and better understand the complex patterns underlying real-world data.

A | Appendices

A.1 Appendix for General and Automated Equivariant Model Construction with Equivariant-MLP

A.1.1 Overview

In subsection A.1.2 we prove that Equation 2.4 and Equation 2.5 are necessary and sufficient conditions to satisfy the symmetry constraint. In subsection A.1.3 we prove that the simple iterative MVM method for finding the nullspace converges to the true nullspace with an exponential rate, and we derive the complexity upper bounds for various symmetry groups and representations. In subsection A.1.4 we show that using gated nonlinearities or Norm-ReLUs alone are not sufficient for universality. In subsection A.1.5 we detail the various groups we implement and calculate the equivariant subspaces for different tensor representations. In subsection A.1.6 we detail the steps necessary to extend our implementation to new groups and representations. subsection A.1.7 details some additional rules by which the solutions for group products can be sped up. Finally, subsection A.1.9 and subsection A.1.10 describe training hyperparameters and how the datasets were constructed.

A.1.2 Necessary and Sufficient Conditions for Equivariance

Theorem A.1. *Given a (real) Lie group G with a finite number of connected components, and a representation ρ acting on vector space V, the symmetry constraint*

$$\forall g \in G : \rho(g)v = v \tag{A.1}$$

for $v \in V$ is satisfied if and only if

$$\forall i = 1, .., D : \qquad d\rho(A_i)v = 0, \tag{A.2}$$

$$\forall \ell = 1, ..., M : \qquad (\rho(h_\ell) - I)v = 0, \tag{A.3}$$

where $\{A_i\}_{i=1}^D$ are D basis vectors for the D dimensional Lie Algebra $\mathfrak{g}$ with induced representation $d\rho$, and for some finite collection $\{h_\ell\}_{\ell=1}^M$ of discrete generators.

Proof: As shown in subsection A.1.8, elements of a (real) Lie group can be written as $g = \exp(\sum_i \alpha_i A_i)\Pi_i h_{k_i}$ for some collection of real valued coefficients $\alpha_i \in \mathbb{R}$ and discrete coefficients $k_i \in [-M, ..., M]$ which index the M discrete generators (and their inverses). Note that M is upper bounded $M \leqslant (D+1) + \mathrm{nc}(G)$, by the sum of the dimension and the number of connected components of G and is often much smaller as shown by the examples in subsection A.1.5. For subgroups of S_n for example, $M \leqslant n$ [Guralnick 1989]. Discrete groups are included as a special case of Lie groups with $D = 0$. The forward and backward directions of the proof are shown below.

Necessary: Assume $\forall g \in G : \rho(g)v = v$. Writing $g = \exp(\sum_i \alpha_i A_i)\Pi_i h_{k_i}$, we can freely choose group elements with $k = \varnothing$ to find that

$$\forall \alpha : \rho\big(\exp(\sum_{i=1}^D \alpha_i A_i)\big)v = \exp\big(d\rho(\sum_{i=1}^D \alpha_i A_i)\big)v = v$$

using the correspondence between group and algebra representation (2.1). From the linearity of

$d\rho(\cdot)$, this implies $\exp\big(\sum_i \alpha_i d\rho(A_i)\big)v = v$. Taking the derivative with respect to α_i at $\alpha = 0$, we have that

$$\forall i = 1, .., D: \quad d\rho(A_i)v = 0, \tag{A.4}$$

since the exponential map satisfies $\frac{d}{dt}\exp(tB)|_{t=0} = B$.

Similarly for the discrete constraints we can set $\alpha_i = 0$ and $k = [\ell]$ so that $\rho(g)v = \rho(\Pi_i h_{k_i})v = \rho(h_\ell)v = v$. By setting varying ℓ, we get the M constraints

$$\forall \ell = 1, ..., M: \quad (\rho(h_\ell) - I)v = 0. \tag{A.5}$$

Sufficient: Assume Equation A.2 and Equation A.3 both hold. Starting with the continuous constraints, exponential map (defined through the Taylor series) satisfies

$$\exp(B)v = v + Bv + \frac{1}{2}B^2v + ...$$

but if $Bv = 0$ then $\exp(B)v = v$ since all terms except v are 0. Setting $B = \sum_i \alpha_i d\rho(A_i) = d\rho(\sum_i \alpha_i A_i)$ which satisfies $Bv = 0$, we have that

$$\begin{aligned} \forall \alpha_i: \quad & \exp\big(d\rho(\sum_i \alpha_i A_i)\big)v = v \\ \forall \alpha_i: \quad & \rho\big(\exp(\sum_i \alpha_i A_i)\big)v = v. \end{aligned} \tag{A.6}$$

Similarly for the discrete generators, if $\forall \ell: \rho(h_\ell)v = v$ then

$$\forall \ell: \quad v = \rho(h_\ell)^{-1}v = \rho(h_{-\ell})v,$$

and any product satisfies

$$\rho(\Pi_{i=1}^{N} h_{k_i})v = \Pi_{i=1}^{N}\rho(h_{k_i})v = \big(\Pi_{i=1}^{N-1}\rho(h_{k_i})\big)\rho(h_{k_N})v.$$

since $\rho(h_{k_N})v = v$ we can remove that factor and repeat the argument to get

$$\rho(\Pi_{i=1}^{N} h_{k_i})v = \big(\Pi_{i=1}^{N-1}\rho(h_{k_i})\big)v = \ldots = v. \tag{A.7}$$

Putting Equation A.6 and Equation A.7 together, we have that

$$\forall \alpha, k : \rho(\exp(\sum_i \alpha_i A_i)\Pi_i h_{k_i})v = v.$$

Since every group element can be expressed as $g = \exp(\sum_i \alpha_i A_i)\Pi_i h_{k_i}$ for some α, k, the equivariance constraint $\forall g \in G : \rho(g)v = v$ is satisfied.

A.1.3 Krylov Method for Efficiently Finding the Equivariant Subspace

The iterative Krylov subspace algorithm that we use to find the nullspace of the constraint matrix C is a close variant of the iterative methods for finding the largest eigenvectors such as power iteration and Ojas method . We need to be able to compute the nullspaces of the massively large constraint matrices C (such as the $(4 \times 10^5) \times (7 \times 10^4)$ sized matrix for computing the equivariant subspace of $T_7^{S_5}$ in subsection A.1.5), making use of efficient structure that allows fast MVMs with C.

While the shift and invert strategy for finding small eigenvalues is commonly recommended [Ipsen 1997], the costs of inversion via conjugate gradients for these massive matrices can make it exceedingly slow in practice. Other methods such as block Lanczos [Eberly 2004] and Wiedemann [Turner 2006] have been explored in the literature for the nullspace problem, but these methods tend to be exceedingly complicated.

Instead we provide a much simpler algorithm that is also extremely fast: simple gradient descent followed by a small SVD. We prove that gradient descent converges exponentially in this problem, and that with probability 1 our method converges to the correct nullspace with error ϵ in $O(\log(1/\epsilon))$ iterations. We then verify this fact empirically. The algorithm is closely related to power iteration [Francis 1961], Oja's rule [Shamir 2015], and the PCA approaches that are framed as optimization problems [Garber and Hazan 2015].

As introduced in algorithm 1 we propose the following algorithm for finding the nullspace with rank $r \leqslant r_{\max}$ and then use iterative doubling of $r_{\max}$ until the conditions are met.

Fast Krylov Nullspace

```
def **CappedKrylovNullspace**($C, r_{\max}$):
$X \sim \mathcal{N}(0,1)^{n \times r_{\max}}$
**while** $L(X) > \epsilon$ **do**
    $L(X) = \|CX\|_F^2$
    $X \leftarrow X - \eta \nabla L$
**end**
$\tilde{Q}, \tilde{\Sigma}, \tilde{V} = \mathrm{SVD}(X)$
**return** $\tilde{Q}$
```

Assume then that $r \leqslant r_{\max}$ (abbreviated r_m) and that we will use the notation from Equation 2.6 that $C = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P^\top \\ Q^\top \end{bmatrix}$. The gradients of L are $\nabla_X L = C^\top C X$ and thus gradient descent can be written as the iteration

$$X_{t+1} = (I - \eta C^\top C) X_t. \tag{A.8}$$

We can write X in terms of the true singular vectors of C (the eigenvectors of $C^\top C$) which form a basis. Q and P are orthogonal ($Q^\top Q = I$, $Q^\top P = 0$, $P^\top P = I$) and we define the projections of X_t onto these subspaces, $W_t = Q^T X_t$ and $V_t = P^T X_t$. As orthogonal transformations of isotropic

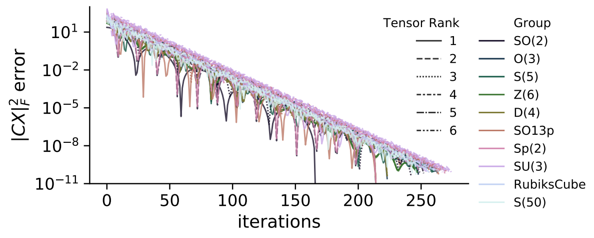


Figure A.1: Exponential convergence of algorithm 1 shown empirically over a range of groups and tensor representations for $r_{\max} = 20$. In each of these cases, X converges to the limits of floating point precision in 300 iterations.

Gaussians are also isotropic Gaussians, the two matrices are initially distributed $W_0 \sim \mathcal{N}(0,1)^{r \times r_m}$ and $V_0 \sim \mathcal{N}(0,1)^{(n-r)\times r_m}$.

Writing $X_t = QW_t + PV_t$, and noting $C^\top C = P\Sigma^2 P^\top$ we can now see the effect of the iteration on the subspaces:

$$\begin{aligned} QW_{t+1} + PV_{t+1} &= (I - \eta P\Sigma^2 P^\top)\big(QW_t + PV_t\big) \\ &= QW_t + P(I - \eta\Sigma^2)V_t. \end{aligned}$$

Unrolling the iteration, we have that $W_t = W_0$ and $V_t = P(I - \eta\Sigma^2)^t V_0$. So long as the learning rate is chosen $\eta < 2/\sigma^2_{\max}$ then the iteration will converge exponentially to $X = QW_0$. Given optimal learning rate, the convergence is $T = O(\kappa \log(1/\epsilon))$ where $\kappa = (\frac{\sigma_{\max}}{\sigma_{\min}})^2$. Since W_0 is a Gaussian random matrix $\mathcal{N}(0,1)^{r\times r_m}$, it will be full rank r with probability 1. Therefore, performing a final SVD on X will yield the nullspace Q. The runtime of this procedure is $O((M+D)\mathcal{T} r_m \log(\frac{1}{\epsilon}) + r_m^2 n)$ since each matrix multiply with C and $C^\top$ takes time $(M+D)\mathcal{T} r_m$ where $\mathcal{T}$ is the time for multiplies with the ρ and $d\rho$ matrices with a single vector, and there are $M + D$ such multiplies that go into a single multiply with C. Finally the $r_m^2 n$ factor is the cost of taking the SVD of X at the end. The exponential convergence is shown empirically across a range of groups in Figure A.1.

If $r > r_{\max}$ then the SVD output $\tilde{Q}$ is a random projection of rank $r_{\max}$ of the true nullspace Q. Given an unknown r, we can simply rerun the algorithm doubling $r_{\max}$ each time until the rank of

$\tilde{Q}$ is less than $r_{\max}$. Adding up the costs, the total runtime of the algorithm to reach an ϵ accurate solution for the nullspace is

$$O((M+D)\mathcal{T} r \log(\tfrac{1}{\epsilon}) + r^2 n). \tag{A.9}$$

To put this runtime into perspective, we can upper bound the runtime to compute the symmetric bases for rank p tensors T_k of any subgroup of the symmetric group $G \leqslant S_m$. Since all $G \leqslant S_m$ can be expressed with $D + M < m$ discrete generators [Guralnick 1989], and axis-wise permutations of the $n = m^k$ sized tensors can be performed in time $\mathcal{T} = m^k$, the runtime is upper bounded by $O(m^k r(m \log(\frac{1}{\epsilon}) + r))$.

Similarly for the orthogonal groups SO(m) and O(m) with $D = m(m-1)/2$ infinitesmal generators and $M \leqslant 1$ discrete generators, the MVM time can be done in $\mathcal{T} = km^k$ since the infinitesmal generators can be expressed with only 2 nonzero elements. Putting this together, the symmetric spaces for rank T_k tensors can be solved for in time $O(m^k r(km^2 \log(\frac{1}{\epsilon}) + r))$, where r is also upper bounded by the Bell numbers $r \leqslant B_k$. Generalizations of the Lorentz group SO$(p, n-p)$ and O$(p, n-p)$ have identical runtimes, and similarly for the complex groups SU(n) and U(n), as well as the symplectic group Sp(n).

For the equivariant maps between the popular *irreducible* representations: $\rho = \psi_k \otimes \psi_\ell$ for the group $G = \mathrm{SO}(3)$, $\mathcal{T} = k^2\ell + k\ell^2$ giving the runtime of our method $O((k^2\ell + k\ell^2) r \log(\frac{1}{\epsilon}) + r^2 k\ell)$. Meanwhile for irreducible representations of $G = \mathrm{SO}(2)$ the runtime is a mere $O(r \log(\frac{1}{\epsilon}) + r^2)$ regardless of k and ℓ.

A.1.4 Linear Layers and Gated Nonlinearities are Not Universal

Outside of the regular representations where each $\rho(g)$ is a permutation matrix, we cannot necessarily use pointwise nonlinearities. Existing equivariant nonlinearities for this setting such as Norm-ReLU and Gated-Nonlinearity can artificially limit the expressivity of the networks in cases such as when using tensor representations.

Theorem A.2. *Consider equivariant networks built only from equivariant linear layers that map between (direct sums of) tensor representations with features $v \in \bigoplus_{a\in\mathcal{A}} T_a$ (as well as biases) and nonlinearities which act separately on each of the tensors $\sigma : T_a \longrightarrow T_a$, or with an additional scalar as $\sigma : T_0 \times T_a \longrightarrow T_a$. There exists groups (like* $\mathrm{SO}(2)$ *and* $\mathrm{O}(3)$*) for which these networks cannot approximate even simple equivariant functions.*

Suppose the base representation ρ of a group G includes elements that satisfy $\exists(g, g') : \rho(g') = -\rho(g)$, which we term the *parity property*. For simplicity assume the representation is orthogonal $\rho = \rho^*$ so that we can talk about rank k tensors rather than rank (p, q) tensors, but the same argument also applies for non-orthogonal representations setting $k = p + q$. Tensor representations $\rho_k(g) = \rho(g)^{\otimes k}$ that have an order k that is *odd* will also have the parity property since $\rho(g')^{\otimes k} = (-1)^k \rho(g)^{\otimes k}$. But because the equivariance constraint holds for all $g \in G$, the following two constraints must also hold for odd k:

$$\rho(g)^{\otimes k} v = v$$
$$-\rho(g)^{\otimes k} v = v.$$

Adding the two constraints together, we have that all equivariant tensors of odd rank (for groups with this property) are $v = 0$. This also means that all equivariant linear maps from a tensor with even rank to a tensor with odd rank will be 0.

This is a property of the equivariance for linear layers for certain groups and is not by itself a problem; however, if the equivariant nonlinearities σ act separately on each tensor and preserve its rank $\sigma : T_a \longrightarrow T_a$ then all quantities in the network (inputs, outputs, and features) of even order are computationally disconnected from those of odd order. Since the nonlinearities act only on a given tensor and keep its order the same, and linear layers between even and odd are 0, there can be no nontrivial path between the two. Similarly for nonlinearities like Gated-Nonlinearities which take in an additional scalar gate as input, so that the nonlinearities are maps $\sigma : T_0 \times T_a \longrightarrow T_a$

we can extend the result. For these kinds of nonlinearities, features of odd order can depend on inputs and features in previous layers of odd and even rank; however, there is still no path from a feature or input of odd order to a later feature or output of even order.

A simple example is the group O(3) and the standard vector representation for $R^\top R = I$, $\rho(R) = R$. Suppose the input to the network is the vector $v \in T_1$ and the target to be learned is the scalar $f(v) = \|v\| \in T_0$. Since the input is an odd order tensor and the output is an even order tensor, there can be no nonzero computational path in the network connecting them. Using Norm-ReLU or Gated-Nonlinearities the only valid output of such a network is a constant c which is independent of the input, and the network cannot fit a simple function such as $\| \cdot \|$.

Of course this limitation extends much beyond this simple example, preventing inner products, matrix vector multiplies, and many other kinds of valid equivariant functions from being expressed, regardless of the size of the network. In fact, using the standard vector representation for all groups $\mathrm{O}(n)$ as well as $\mathrm{SO}(2n)$ satisfies the parity property, and will provably have this limitation. Other groups like the Lorentz group $\mathrm{SO}(1,3)$ and $\mathrm{O}(1,3)$, and the symplectic group $\mathrm{Sp}(n)$ satisfy this property.

A.1.5 Equivariant Bases for Various Groups

In this section we list the dimension of the symmetric bases for various groups and tensor representations that we calculate using our algorithm, and visualize the bases.

A.1.5.1 Discrete Translation Group $\mathbb{Z}_n$

The discrete translation group (or cyclic group) $\mathbb{Z}_n$ is generated by a single shift permutation $\mathrm{P}[n, 1, 2, ..., n-1]$. Translation equivariant neural networks make use of convolutions which are maps $(T_1 \longrightarrow T_1) \cong T_2$ and average pooling $(T_1 \longrightarrow T_0) \cong T_1$. We recover the n dimensional convolution bases and the 1 dimensional average pooling element as shown in the table and Figure 2.3. We also show the ranks for higher order tensors, which appear to satisfy $r = n^{k-1}$ for

$\mathbb{Z}_n$ and T_k. While we are not aware of the higher order equivariant tensor having been derived in the literature, it's not unlikely due to the prominence of the translation group in signal processing.

	$\mathbb{Z}_2$	$\mathbb{Z}_3$	$\mathbb{Z}_4$	$\mathbb{Z}_5$	$\mathbb{Z}_6$	$\mathbb{Z}_7$	$\mathbb{Z}_8$
T_1	1	1	1	1	1	1	1
T_2	2	3	4	5	6	7	8
T_3	4	9	16	25	36	49	64
T_4	8	27	64	125	216	343	512
T_5	16	81	256	625	1296	2401	
T_6	32	243	1024	3125			
T_7	64	729	4096				

Table A.1: Symmetric subspace rank r for tensors T_k of $G = \mathbb{Z}_n$

A.1.5.2 Permutation Group $\mathbb{S}_n$

We review the solutions to the permutation group $\mathbb{S}_n$ which were solved for analytically in Maron et al. [2018], which we solve for numerically using our algorithm. As expected, the solutions bases match the limiting size of the kth Bell number B_k as $n \to \infty$.

However, Maron et al. [2018] claim that the size of the basis is always B_k regardless of n and that is not quite correct. The basis derived in Maron et al. [2018] is always equivariant, but sometimes it contains linearly dependent solutions, leading to an overcounting when n is small. The fact that the basis cannot always be of size B_k can be seen from the fact that the total dimension of T_k is n^k and the equivariant subspace thus has rank $r \leqslant n^k$. The Bell numbers grow super exponentially in k (about $(k/\log(k))^k$) and therefore given any n they must exceed the maximum n^k for some value of k. The place where the original argument of Maron et al. [2018] breaks down is when the equivalence classes γ may be empty. For example the equivalence class $\gamma_1 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$ corresponds to indices i_1,i_2,i_3,i_4 which are all distinct. But for $n < 4$ one cannot form a set of four indices that are all distinct, hence γ_1 is empty for $n < 4$. To the best of our knowledge, the dimension of the equivariant subspaces for small n that we report below are the precise values and have not been presented anywhere else.

	$\mathbb{S}_2$	$\mathbb{S}_3$	$\mathbb{S}_4$	$\mathbb{S}_5$	$\mathbb{S}_6$	$\mathbb{S}_7$	$\mathbb{S}_8$	B_k
T_1	1	1	1	1	1	1	1	1
T_2	2	2	2	2	2	2	2	2
T_3	4	5	5	5	5	5	5	5
T_4	8	14	15	15	15	15	15	15
T_5	16	41	51	52	52	52	52	52
T_6	32	122	187	202	203	203		203
T_7	64	365	715	855				877

Table A.2: Symmetric Subspace rank r for tensors T_k of $G = \mathbb{S}_n$

A.1.5.3 Rubik's Cube Group

Showing the capabilities to apply to unexplored groups and representations, we compute the equivariant bases for linear layers that are equivariant to the action of the Rubik's Cube group. The Rubik's cube group is a subgroup of the permutation group $G < S_{48}$ containing all valid Rubik's cube transformations. The group is extremely large $|G| > 4 \times 10^{19}$, but is generated by only 6 generators: F, B, U, D, L, R a quarter turn about the front, back, up, down, left, and right faces.

We use the standard 48 dimensional regular representation where each of the $6 * (9 - 1) = 48$ facets (the center facets are excluded) is a component. The state of the Rubik's cube is represented by a 48 dimensional vector where each component is an integer $0 - 5$ representing the 6 possible colors each facet can take. Using this representation, the 6 generators can be expressed as permutations and we refer the readers to the code for the (lengthy) values of the permutations.

Below we show the dimension of the equivariant basis and the size of the tensors in which the basis is embedded. Note that as the Rubik's cube is a subgroup of S_{48}, and has fewer group elements as symmetries, the size of the equivariant basis is larger $2, 6, 22, \ldots$ vs $1, 2, 5, \ldots$. For T_4 of size $48^4 = 5308416$ we were able to run the solver with $r_{\max} = 20$ before running out of GPU memory.

	T_1	T_2	T_3	T_4
r	2	6	22	>20
48^k	48	2304	110592	5308416

Table A.3: Symmetric Subspace rank r for tensors T_k of Rubik's Cube Group

A.1.5.4 Continuous Rotation Groups SO(n) and O(n)

The special orthogonal group SO(n) and the orthogonal group O(n) are continuous Lie groups have the Lie algebra

$$\mathfrak{o}(n) = \mathfrak{so}(n) = T_{\mathrm{id}}\mathrm{SO}(n) = \{A \in \mathbb{R}^{n\times n} : A^\top = -A\}$$

of dimension $D = n(n-1)/2$. The orthogonal group can be constructed with the additional discrete generator $h = \begin{bmatrix} -1 & 0 \\ 0 & I_{n-1} \end{bmatrix}$ that has $\det(h) = -1$.

	SO(2)	SO(3)	SO(4)	SO(5)	SO(6)	SO(7)
T_2	2	1	1	1	1	1
T_3	0	1	0	0	0	0
T_4	6	3	4	3	3	3
T_5	0	6	0	1	0	0
T_6	20	15	25	15	16	15
T_7	0	36	0	15		
T_8	70	91	196			

Table A.4: Symmetric subspace rank r for tensors T_k of $G = \mathrm{SO}(n)$

We omit the first row T_1 since all the values are 0. The additional basis element along the diagonal $n = k$ can be recognized as the well known anti-symmetric Levi-Civita symbol $\varepsilon_{ijk\ell\ldots}$. However this basis element does not respect orientation reversing isometries, and is thus absent in the equivariant basis for $\mathrm{O}(n)$:

	O(2)	O(3)	O(4)	O(5)	O(6)	O(7)
T_2	1	1	1	1	1	1
T_3	0	0	0	0	0	0
T_4	3	3	3	3	3	3
T_5	0	0	0	0	0	0
T_6	10	15	15	15	15	15
T_7	0	0	0	0		
T_8	35	91	105			

Table A.5: Symmetric subspace rank r for tensors T_k of $G = \mathrm{O}(n)$

A.1.5.5 Lorentz Groups $\mathrm{SO}^+(1,3)$, $\mathrm{SO}(1,3)$, $\mathrm{O}(1,3)$

The Lorentz group is defined as the set of matrices that preserve the Lorentz metric η: $\mathrm{O}(1,3) = \{L \in \mathbb{R}^{4\times 4} : L^\top \eta L = \eta\}$. Differentiating, one gets the $D = 6$ dimensional Lie algebra $\mathfrak{so}(1,3) = \{A \in \mathbb{R}^{4\times 4} : A^\top \eta + \eta A = 0\}$. The full Lorentz group $\mathrm{O}(1,3)$ has four connected components.

The identity component of the Lorentz group $\mathrm{SO}^+(1,3)$ is just the exponential of the Lie algebra $\mathrm{SO}^+(1,3) = \exp(\mathfrak{o}(1,3))$. The subgroup $\mathrm{SO}(1,3)$ of $\mathrm{O}(1,3)$ with determinant 1 can be constructed with the additional generator $h_1 = -I$ (which combines time reversal with a parity transformation), and the full Lorentz group $\mathrm{O}(1,3)$ includes h_1 as well as the generator $h_2 = \begin{bmatrix} -1 & 0 \\ 0 & I_3 \end{bmatrix}$ that reverses time only. As these groups are not orthogonal, we must distinguish $T_{(a,b)}$ from $T_{(a+b,0)}$. Below we show the number of basis vectors for $T_{(k,0)}$ which for these 3 groups is the same number as for $T_{(k-i,i)}$ although the bases elements are distinct.

	$T_{(2,0)}$	$T_{(3,0)}$	$T_{(4,0)}$	$T_{(5,0)}$	$T_{(6,0)}$	$T_{(7,0)}$	$T_{(8,0)}$
$\mathrm{SO}^+(1,3)$	1	0	4	0	25	0	196
$\mathrm{SO}(1,3)$	1	0	4	0	25	0	196
$\mathrm{O}(1,3)$	1	0	3	0	15	0	105

Table A.6: Symmetric subspace rank r for tensors $T_{(k,0)}$ for the Lorentz groups.

A.1.5.6 Symplectic Group $\mathrm{Sp}(n)$

Similar to the orthogonal group and the Lorentz group, the symplectic group is defined through the perservation of a quadratic form.

$$\mathrm{Sp}(n) = \{M \in \mathbb{R}^{2n\times 2n} : M^\top \Omega M = \Omega\},$$

where $\Omega = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$ and is often relevant in the context of Hamiltonian mechanics and classical physics. The quadratic form Ω can be interpreted as a measurement of oriented area (in phase space) and is preserved by the evolution of many systems. The $D = n(2n+1)$ dimensional Lie algebra satisfies

$$\mathfrak{sp}(n) = \{A \in \mathbb{R}^{2n\times 2n} : A^\top\Omega + \Omega A = 0\},$$

and any element in $\mathrm{Sp}(n)$ can be written $M = \exp(A_1)\exp(A_2)$ for some $A_1, A_2 \in \mathfrak{sp}(n)$.

	Sp(1)	Sp(2)	Sp(3)	Sp(4)	Sp(5)	Sp(6)
$T_{(2,0)}$	1	1	1	1	1	1
$T_{(3,0)}$	0	0	0	0	0	0
$T_{(4,0)}$	2	3	3	3	3	3
$T_{(5,0)}$	0	0	0	0	0	0
$T_{(6,0)}$	5	14	15	15		
$T_{(7,0)}$	0	0	0			
$T_{(8,0)}$	14	84				

Table A.7: Symmetric subspace rank r for tensors $T_{(k,0)}$ of $G = \mathrm{Sp}(n)$

Although for large values of n, the dimension of the basis for $\mathrm{Sp}(n)$ becomes similar to that of its subgroup $\mathrm{O}(n)$, the basis elements themselves are quite different. Like the Lorentz groups, different ways of distributing the rank between the base vector space and its dual as $T_{(k-i,i)}$ for different i yields different solutions for the equivariant basis.

A.1.5.7 Special Unitary Group SU(n)

Using a complex valued SVD and replacing the objective in the iterative algorithm $L(Q) = \|CQ\|_F^2 = \mathrm{Tr}(Q^\top C^\top CQ)$ with $L(Q) = \mathrm{Tr}(Q^\dagger C^\dagger CQ)$ where $\dagger$ is the complex conjugate transpose, we can apply our method to solve for the equivariant bases for complex groups such as the special unitary group $\mathrm{SU}(n)$ relevant for the symmetries of the standard model of particle physics.

The group can be defined

$$\mathrm{SU}(n) = \{U \in \mathbb{C}^{n\times n} : U^\dagger U = 1, \ \ \det(U) = 1\}.$$

The Lie algebra of dimension $D = n^2 - 1$ satisfies

$$\mathfrak{su}(n) = \{A \in \mathbb{C}^{n\times n} : A^\dagger = -A, \ \ \mathrm{Tr}(A) = 0\}.$$

The group is contained in the image of the exponential map, $\exp(\mathfrak{su}(n)) = \mathrm{SU}(n)$. Since the size of the basis differs between $T_{(4,1)}$ and $T_{(3,2)}$ for example, we show the solutions for only a selection of tensor ranks. It may also be useful to consider *anti*-linear maps, but we leave this to future work.

	$T_{(3,0)}$	$T_{(3,1)}$	$T_{(3,2)}$	$T_{(3,3)}$	$T_{(4,0)}$	$T_{(4,1)}$	$T_{(4,2)}$
SU(2)	0	2	0	5	2	0	5
SU(3)	1	0	0	6	0	3	0
SU(4)	0	0	0	6	1	0	0

Table A.8: Symmetric subspace rank r for tensors $T_{(q,p)}$ of $G = \mathrm{SU}(n)$

A.1.6 Recipe for Use

Below we outline the minimum required steps for adding new groups and representations to our existing implementation written in Jax [Bradbury et al. 2018].

Adding new groups:

1. Specify a sufficient set of M discrete generators and their base representation as a matrix

$\rho(h_i)$ or as a matrix vector multiply $v \longrightarrow \rho(h_i)v$ for each $i = 1, 2, ..., M$

2. Specify a basis for the Lie algebra (if any) and its base representation as a matrix $d\rho(A_i)$ or as a matrix vector multiply $v \longrightarrow d\rho(A_i)v$ for each $i = 1, 2, ..., D$

We walk through these steps and provide examples in **Adding new representations $\tilde{\rho}$ to existing groups**:

1. Specify $\tilde{\rho}(h_i)$ as a function of $\rho(h_i)$
2. Define $\dim(V)$, ==, and hash functions for the representation

We provide more detailed instructions along with examples for implementing new groups at https://emlp.readthedocs.io/en/latest/notebooks/3new_groups.html and new representations at https://emlp.readthedocs.io/en/latest/notebooks/4new_representations.html.

Given that a base representation ρ is *faithful*, all representations can be constructed as functions of this ρ whatever it may be.

A specification of $\tilde{\rho}(h) = f(\rho(h))$ induces the Lie algebra representation $d\tilde{\rho}(A)$ which may be computed automatically using autograd Jacobian vector products (JVP), without needing to specify it manually. $d\tilde{\rho}(A) = \mathrm{JVP}(f, I, d\rho(A))$ which corresponds in math terms to the operation $d\tilde{\rho}(A) = Df|_{\rho(h)=I}(d\rho(A))$.

A.1.7 Group Products

Many of the relevant groups for larger problems like 2D arrays, GCNNs, point clouds [Fuchs et al. 2020], sets of images [Maron et al. 2020], and hierarchical structures [Wang et al. 2020] have multiple distinct group substructures. 2D translation symmetry is the group $G = \mathbb{Z}_n \times \mathbb{Z}_n = \mathbb{Z}_n^2$, GCNNs have $G = H \ltimes \mathbb{Z}_n^2$ point clouds typically have $G = S_n \times \mathrm{E}(3)$, sets of images have the symmetry $S_m \times \mathbb{Z}_n^2$, and a voxelized point cloud network could be $(S_n \times \mathrm{E}(3)) \wr \mathbb{Z}_m^3$. Here these symbols for combining groups are the direct product ($\times$), semi-direct product ($\ltimes$) and wreath product ($\wr$). An additional asymptotic speedup can be achieved for groups that are constructed using these structures by exploiting knowledge about how solutions for the larger group depend on the solutions for the constituent groups.

Suppose representation ρ_a of the group G_a acts on the space V_a which has the symmetric basis Q_a, and ρ_b of G_b acts on V_b with the symmetric basis Q_b.

($\times$): As shown in [Maron et al. 2020], the equivariant basis for $G_a \times G_b$ with rep $\rho_a \otimes \rho_b$ can be written as the Kronecker product $Q_{ab} = Q_a \otimes Q_b$.

($\wr$): In [Wang et al. 2020] it was worked out that the equivariant basis for $G_a \wr G_b$ with rep $(\rho_a \wr \rho_b) \otimes (\rho_a \wr \rho_b)^*$ satisfies $\mathrm{unvec}(Q_{ab}[\alpha, \beta]) = \mathrm{unvec}(Q_a\beta) \otimes I + \mathbb{1}\mathbb{1}^\top \otimes \mathrm{unvec}(Q_b\alpha)$ where α, β are coefficient vectors of size r_a,r_b and the $\mathrm{unvec}(\cdot)$ operation reshapes a vector into a matrix.

A.1.8 Parametrizing Lie Groups with Continuous and Discrete Generators

According to Winkelmann [2003], every real connected Lie group G of dimension D contains a dense subgroup $H \leqslant G$ generated by $D + 1$ elements (which can be constructed explicitly by sampling elements in a neighborhood of the identity). Since H is dense in G, for every element $g \in G$ has a neighborhood $\mathcal{N}(g)$ which contains an element $h \in H$. Since $h \in \mathcal{N}(g)$ it must also be true that $gh^{-1} \in \mathcal{N}(\mathrm{id})$ is in a neighborhood of the identity. Since the exponential map exp is a bijection for a neighborhood around the identity, it must be possible to express $gh^{-1} = \exp(A) = \exp(\sum_i \alpha_i A_i)$ for some $A \in \mathfrak{g}$ that we can then write in terms of the basis elements A_i. Rearranging, and expressing h in terms of the finite generators for H, $h = \Pi_i h_{k_i}$, we have that any $g \in G$ can be written

$$g = \exp(\sum_i \alpha_i A_i)\Pi_i h_{k_i}. \tag{A.10}$$

or more succinctly $G = \exp(\mathfrak{g})H$. It is likely that the result can also be extended to other fields like $\mathbb{C}$ such as by embedding the group in a higher dimensional real group, but we focus on the real case here. For groups with multiple connected components, we can apply the same result with at most one additional discrete generator for each of the connected components.

A.1.9 Implementation Details

While there is substantial freedom in the chosen representation for a given feature layer of a neural network, for maximum expressiveness in the subsequent neural network architecture given a fixed channel budget, we suggest allocating the channels to the different representations in each layer with a uniform allocation heuristic. Progressing from lower dimensional representations to higher dimensional ones, the multiplicity of the representation should be chosen so that the number of channels for associated with each is approximately the same. We use this heuristic for the equivariant networks of each experiment in the paper.

For the three synthetic experiments, we use networks constructed with 3 EMLP layers each with $c = 384$ channels, followed by a single equivariant linear layer mapping to the output type. For the bilinear layer, The baseline MLPs also have 3 hidden layers of size $c = 384$ each. We train all models with batchsize 500, and learning rate 3×10^{-3} with the Adam optimizer [Kingma and Ba 2014]. We train for a total of $\min(900000/N, 1000)$ epochs where N is the size of the dataset, which we found was ample for convergence of both models in all cases. The training time for the EMLP is a couple minutes, while the MLP model trains in < 1 minute.

For the modeling of the double spring pendulum dynamical system we use the same hyperparameters except with $c = 128$ for all models. For the numerical integrator, we use the adaptive RK integrator that is default to Jax with tolerance 2×10^{-6}. We measure relative error as $\text{relative\_error}(a, b) = \|a - b\|/(\|a\| + \|b\|)$. For calculating state relative error, the state is the full vector z of both position and momentum. We train for a total of 2000 epochs, long enough for each of the models to converge.

A.1.10 Datasets

We generate the O(5) invariant dataset using the function $f(x_1, x_2) = \sin(\|x_1\|) - \|x_2\|^3/2 + \frac{x_1^\top x_2}{\|x_1\|\|x_2\|}$ and sampling $x_1, x_2 \sim \mathcal{N}(0,1)^5$. We generate the O(3) equivariant inertia dataset by computing $\mathcal{I} = \sum_{i=1}^5 m_i(x_i^\top x_i I - x_i x_i^\top)$ for $x_i \sim \mathcal{N}(0,1)^3$ and sampling positive masses by passing random entries through a softplus: $m_i \sim \mathrm{Softplus}(\mathcal{N}(0,1))$.

For the Lorentz invariant particle interaction dataset we calculate the targets $y = 4[p^{(\mu}\tilde{p}^{\nu)} - (p^\alpha\tilde{p}_\alpha - p^\alpha p_\alpha)\eta^{\mu\nu}][q_{(\mu}\tilde{q}_{\nu)} - (q^\alpha\tilde{q}_\alpha - q^\alpha q_\alpha)\eta_{\mu\nu}]$ from the sampled momenta $p_\mu, \tilde{p}_\mu, q_\mu, \tilde{q}_\mu \sim \mathcal{N}(0, 1/4^2)$.

For each of these datasets we separate out a test set of size 5000 and a validation set of size 1000 which we use for early stopping.

For the double spring dynamical system, we generate the ground truth trajectories using the Hamiltonian dynamics of the Hamiltonian

$$H(x_1, x_2, p_1, p_2) = V(x_1, x_2) + T(p_1, p_2)$$

where $T(p_1, p_2) = \|p_1\|^2/2m_1 + \|p_2\|^2/2m_2$ and $V(x_1, x_2) =$

$$\tfrac{1}{2}k_1(\|x_1\| - \ell_1)^2 + \tfrac{1}{2}k_2(\|x_1 - x_2\| - \ell_2)^2 + m_1 g^\top x_1 + m_2 g^\top x_2.$$

The constants are chosen $m_1 = m_2 = k_1 = k_2 = \ell_1 = \ell_2 = 1$. The gravity direction is down $g = [0,0,1]$. We sample the initial conditions from the distribution $x_1 \sim [0,0,-1.5] + \mathcal{N}(0, .2^2)^3$, $x_2 \sim [0,0,-3] + \mathcal{N}(0, .2^2)^3$ and $p_1, p_2 \sim \mathcal{N}(0, .4^2)^3$. We integrate these systems for a time $T = 30s$ and for each initial condition we select a randomly chosen $1s$ chunk (evaluated at five $0.2s$ intervals) as the training data. We generate 1500 trajectory chunks which we split up into 500 for each of the train, validation, and test sets.

A.2 Appendix for Residual Pathway Priors for Approximate Equivariance

Appendix Outline

In Section 3.7 discuss potential for negative impact. In Section A.2.2 we investigate the utility of using RPP-EMLP for the policy function only on the Mujoco tasks. In Section A.2.3 we detail the datasets and experimental methodology used in the paper. Finally in Sections A.2.4 and A.2.5 we break down the components of the Mujoco environment state and action spaces, and the representations that we use for them.

A.2.1 Potential Negative Impacts

As one of our primary application areas is reinforcement learning, and specifically exploiting approximate symmetries in reinforcement learning, we must address the potential negative impacts of the deployment of RPPs in RL systems. In general model free RL algorithms tend to be brittle, and often policies and behavior learned in a simulated environment like Mujoco don't transfer easily to real world robots. This point is acknowledged by most RL researchers, and a large effort is being made to improve the situation. Applying neural networks to the control of real robots can be dangerous if the functions are important or failure can cause injury to the robot or humans. We believe that RL will ultimately be impactful for robot control, however practitioners need to be responsible and exercise caution.

A.2.2 Benefit of Equivariant Value Functions

In principle both the policy and the value or critic function can benefit from equivariance. However, the policy learns from the value function in the policy update which is approximately equivalent

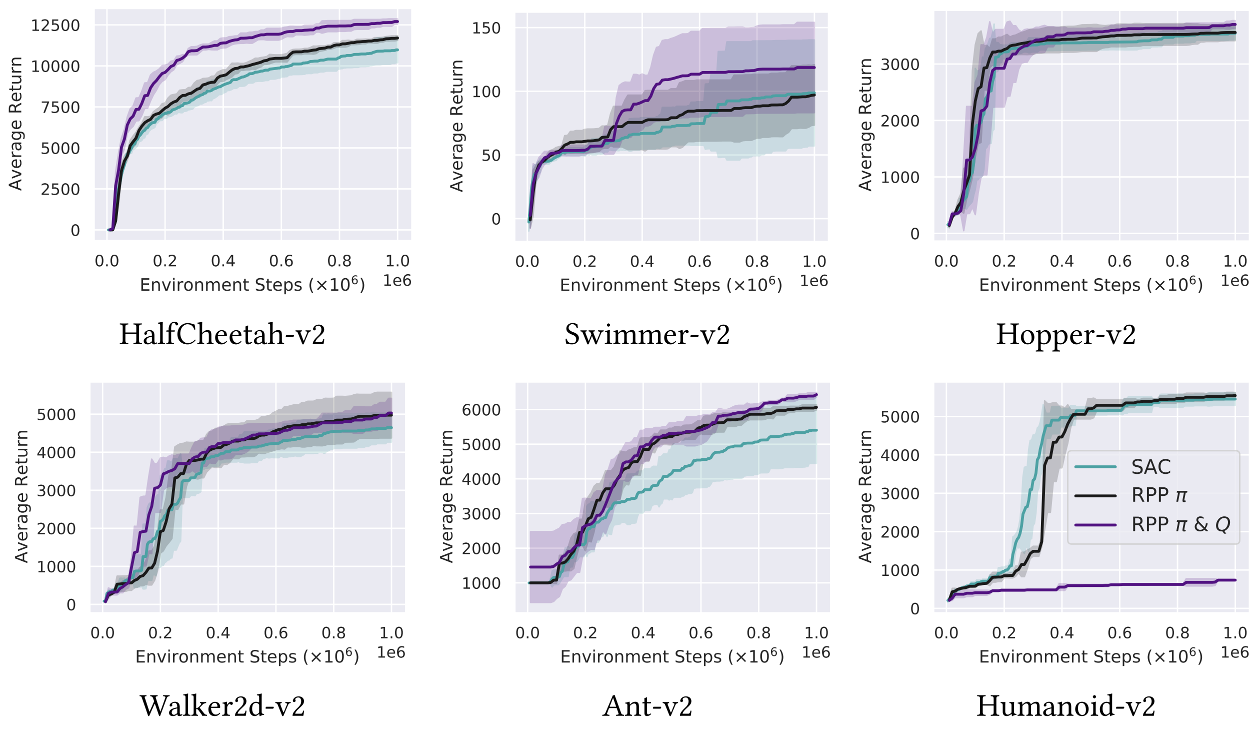


Figure A.2: Average reward curves (max over steps) for an RPP-EMLP applied to the policy π only, as well as an RPP-EMLP for both the policy π and the critic Q. Mean and standard deviation taken over 4 trials shown in the shaded region. Only minor performance gains are achieved if using RPP for the policy only, however this variant is more stable and can to train on Humanoid-v2 without diverging.

to minimizing the KL divergence

$$\mathbb{E}_{s\sim\mathcal{D}}[\mathrm{KL}(\pi_\phi(\cdot|s)|\exp(Q_\theta(\cdot,s))/Z_\theta(s))]$$

as derived in Haarnoja et al. [2018b]. If the value function Q is a standard MLP yielding a non equivariant distribution and the policy function π is an RPP that merely has a bias towards equivariance, then the RPP policy will learn to fit the non equivariant parts of Q as if it were a ground truth dataset that is not equivariant. This likely explains why we find in practice that using an RPP for the value function has a stronger impact on performance as shown in Figure 3.4.

A.2.3 Experimental Details

Here we present the training details of the models used in the paper. Experiments were run on private servers with NVIDIA Titan RTX and RTX 2080 Ti GPUs. We estimate that all runs performed in the initial experimentation and final evaluation on the RL tasks used approximately 500 GPU hours. The experiments on dynamical systems, CIFAR-10, and UCI data required an additional 200 GPU hours.

A.2.3.1 Synthetic Dataset Experiments (3.5.1 and 3.5.2)

The windy pendulum dataset is a variant of the double spring pendulum Hamiltonian system from Finzi et al. [2021]. In addition to the Hamiltonian of the base system

$$H_0(x_1, x_2, p_1, p_2) = V(x_1, x_2) + T(p_1, p_2)$$

where $T(p_1, p_2) = \|p_1\|^2/2m_1 + \|p_2\|^2/2m_2$ and $V(x_1, x_2) =$

$$\tfrac{1}{2}k_1(\|x_1\| - \ell_1)^2 + \tfrac{1}{2}k_2(\|x_1 - x_2\| - \ell_2)^2 + m_1 g^\top x_1 + m_2 g^\top x_2,$$

we add a perturbation $H_1(x_1, x_2, p_1, p_2) = -w^\top x_1 - w^\top x_2$ that is the energy of the wind acting as a constant force pushing in the $w = [-8, -5, 0]$ direction. Setting $H = H_0 + \epsilon H_1$, we can control the strength of the wind and we choose $\epsilon = 0.01$. This perturbation breaks the SO(2) symmetry about the z axis.

For the MLP, EMLP, and RPP we use 3 layer deep 128 hidden unit Hamiltonian neural networks [Greydanus et al. 2019] to fit the data using the rollouts of an ODE integrator [Chen et al. 2018] with an MSE loss on rollouts of length 5 timesteps with $\Delta t = 0.2$. For training we use 500 trajectory chunks and use another 500 for testing. We train all models in section 3.5.1 for 1000 epochs, sufficient for convergence. The input and output representation for EMLP and RPP-EMLP is

$V^4_{\mathrm{O}(3)} \to \mathbb{R}$, where $V_{\mathrm{O}(3)}$ is the restricted representation from the standard representation of a 3D rotation matrix to the given group in question, like $\mathrm{SO}(2)$ for rotations about the z axis. The input is $V^4_{\mathrm{O}(3)}$ because there are two point masses each of which has a 3D vectors for position and for momentum. The scalar $\mathbb{R}$ output is the Hamiltonian function.

The Modified Inertia dataset is a small regression dataset off of the task also from Finzi et al. [2021] for learning the moment of inertia matrix in 3D of a collection of 5 point masses. For the base Inertia dataset, the targets are $\mathcal{I} = \sum_{i=1}^{5} m_i(x_i^\top x_i I - x_i x_i^\top)$ from the input tuples $(m_i, x_i)_{i=1}^{5}$. In order to break the equivariance of the dataset, we add an additional term so that the target is $y = \mathrm{vec}(\mathcal{I} + 0.3\mathcal{I}^2 \hat{z}\hat{z}^\top \mathcal{I})$ where $\hat{z}$ is the unit vector along the z axis. The input and output representations for EMLP and RPP-EMLP on this problem are $(\mathbb{R} \oplus V)^5 \to V \otimes V$, representing the 5 point masses and vectors mapping to matrices $V \otimes V$.

We use 1000 train and test examples for the inertia datasets and we train for 500 epochs. In both cases we use an Adam optimizer [Kingma and Ba 2014] with a learning rate of 0.003.

A.2.3.2 Image and UCI experiments (3.5.3)

We use the CIFAR-10 and UCI datasets, taken from Krizhevsky et al. [2009] and Dua and Graff [2017] respectively. In Section 3.5 we train models on dynamical systems and CIFAR-10 and UCI regression data. For the CIFAR-10 experiments we use a convolutional neural network (and the equivalent MLP) with 9 convolutional layers and 1 fully connected layer, and max-pooling layers after the third and sixth convolutional layers. The channel sizes of the 9 layers are, in order: 16, 16, 16, 32, 32, 32, 32, 32, 32. We train for 200 epochs using a cosine learning rate schedule with an initial learning rate of 0.05 and the Adam optimizer.

For the UCI tasks we use a small convolutional neural network, and the equivalent MLP, with 3 convolutional layers and 1 fully connected layer, with each convolutional layer having 32 channels. Models are trained for 1000 epochs using an Adam optimizer with a learning rate of 0.01 and a cosine learning rate schedule.

A.2.3.3 Model Free RL

We train on the Mujoco locomotion tasks in the OpenAI gym environments [Brockman et al. 2016]. We follow the implementation details and hyperparameters from Haarnoja et al. [2018c], with a learned temperature function, stochastic policies, and double critics. Additionally we use the recommendation from Andrychowicz et al. [2020] to initialize the last layer of the policy network with 100x smaller weights, which we find slightly improves the performance of both RPP and the baseline. Additionally for RPP which can be less stable than standard SAC, we use the Adam betas $\beta_1 = 0.5$ and $\beta_2 = 0.999$ that are used in he GAN community [Miyato et al. 2018] rather than the defaults. Training with the RPP π and Q functions on the Mujoco locomotion tasks takes about 8 hours for 1 million steps.

We found it necessary to reduce the speed τ of the critic moving average to keep SAC stable on some of the environments, with values shown in Table A.9. In general, higher τ's are favorable for learning quickly. Unfortunately we were not able to get SAC with an RPP Q function to train reliably on Humanoid, even after trying multiple values of τ.

	Walker2d	Hopper	HalfCheetah	Swimmer	Ant	Humanoid
Baseline τ	.005	.005	.005	.005	.005	.005
RPP τ	.004	.005	.005	.004	.005	✗

Table A.9: Critic moving average speed τ.

A.2.3.4 Transition Models for Mujoco

We train the transition models on a dataset of 50000 transitions which are composed of 5000 trajectory chunks of length 10. These trajectory chunks are sampled uniformly from the replay buffer collected over the course of training a standard SAC agent for 10^6 steps on each of the environments. We train by minimizing the $\ell 1$ norm of the rollout error over a 10 step trajectory, and we evaluate on a holdout set of 50 trajectories of length 100.

The models are simple MLPs or RPPs mapping from the state and control actions to the state space, predicting the change in state,

$$x_{t+1} = x_t + \mathrm{NN}(x_t, u_t).$$

For the MLPs and RPPs we use 2 hidden layers of size 256 as well as swish activations [Ramachandran et al. 2017]. We use a prior variance of 10^6 in the equivariant subspace and 3 in the non equivariant subspace. The RPP is a standard RPP-EMLP with the input representation $\rho_X \oplus \rho_U$ (concatenation of the representation of the state space and the action space), output representation ρ_X, and symmetry group described in subsection A.2.4 the same as for the model free experiments. We train the transition models for 500 epochs which takes about 45 minutes for RPP compared to 15 minutes for the standard MLPs.

A.2.4 Mujoco State and Action Representations

Based on the state and action spaces of the Mujoco environments we describe in subsection A.2.5, we define appropriate group representations on these spaces. Let V be the base representation of the group acted upon by permutations for $\mathbb{Z}_n$ and by rotation matrices for $\mathrm{SO}(2)$, let $\mathbb{R}$ denote a scalar representation (of dimension 1) that is unaffected by the transformations, and let P be a pseudoscalar representation (of dimension 1) that transforms by the sign of the permutation. For $\mathbb{Z}_2$, P takes the values 1 and -1 and acts by negating the values when a flip or L/R reflection is applied.

From the raw state and action spaces listed in subsection A.2.5, we convert quaternions to 3D rotation matrices for Humanoid and Ant, and we reorder elements to group together left/right pairs for Walker2d and Swimmer. The representations of these transformed state and action vectors are shown in Table A.10. Note that V^3 denotes $V \oplus V \oplus V = V^{\oplus 3}$, and is simply the concatenation of 3 copies of V as $\mathbb{R}^3$ would be 3 copies of $\mathbb{R}$. This is not to be confused with powers

Table A.10: Mujoco Locomotion State and Action Representations used for RPP-EMLP

Env	State Representation	Action Rep	Group
Hopper	$\mathbb{R} \oplus P^5 \oplus \mathbb{R} \oplus P^4$	P^3	$\mathbb{Z}_2$
Swimmer	$\mathbb{R} \oplus P_{\leftrightarrow} \oplus (P_{\leftrightarrow} \otimes V_{\updownarrow}) \oplus (\mathbb{R} \oplus P)^2 \oplus (P_{\leftrightarrow} \otimes V_{\updownarrow})$	$P_{\leftrightarrow} \otimes V_{\updownarrow}$	$\mathbb{Z}_2^{\leftrightarrow} \times \mathbb{Z}_2^{\updownarrow}$
HalfCheetah	$\mathbb{R} \oplus P^8 \oplus \mathbb{R} \oplus P^7$	P^6	$\mathbb{Z}_2$
Walker2d	$\mathbb{R}^2 \oplus V^3 \oplus \mathbb{R}^3 \oplus V^3$	V^3	$\mathbb{Z}_2$
Ant	$\mathbb{R}^5 \oplus V^2 \oplus \mathbb{R}^6 \oplus V^2$	V^2	$\mathbb{Z}_4$
Humanoid	$\mathbb{R} \oplus V_{\mathrm{SO}(3)}^{\otimes 2} \oplus \mathbb{R}^{17} \oplus V_{\mathrm{SO}(3)}^2 \oplus \mathbb{R}^{17}$	$\mathbb{R}^{17}$	$\mathrm{SO}(2)$

of the tensor product, $V^{\otimes 3} = V \otimes V \otimes V$. For Humanoid, we denote the restricted representation of 3D rotation matrices restricted to the SO(2) rotations about the z axis as $V_{\mathrm{SO}(3)}$.

A.2.5 Mujoco State and Action Spaces

In order to build symmetries into the state and action representations for Mujoco environments, we need to have a detailed understanding of what the state and action spaces for these environments represent. As these spaces are not well documented, for each of the Mujoco environments we experimented in the simulator and identified the meanings of the state vectors in Tables A.15, A.17, A.16, A.12, A.14, A.11, and A.13. We hope that these detailed descriptions can be useful to other researchers.

Table A.11: Hopper-v2 State and Action Spaces

State Space	X (Unobserved)
	Y
	Orientation Angle
	Hip Angle
	Knee Angle
	Ankle Angle
	X Velocity
	Y Velocity
	Orientation Angular Velocity
	Hip Angular Velocity
	Knee Angular Velocity
	Ankle Angular Velocity
Action Space	Hip
	Knee
	Ankle

Table A.12: Swimmer-v2 State and Action Spaces

State Space	X (Unobserved)
	Y (Unobserved)
	Orientation Angle
	Head Joint Angle
	Tail Joint Angle
	X Velocity
	Y Velocity
	Orientation Angular Velocity
	Head Joint Angular Velocity
	Tail Joint Angular Velocity
Action Space	Head Joint
	Tail Joint

A.3 Appendix for Lie Derivative for Measuring Learned Equivariance

A.3.1 Lie Groups, Lie Derivatives, and LEE

A.3.1.1 Lie Groups and Local/Global Notions of Equivariance

The key to understanding why the local - global equivalence holds is that $(\exp(X)-1) = \sum_{k=1}^{\infty} X^k/k!$ has the same nullspace as X (here repeated application of X on a function f is just the repeated directional derivative, and this is the definition of a vector field used in differential geometry). Since they have the same nullspace, the space of functions for which $\exp(X)f = f$ is the same as the space $Xf = 0$. The same principle holds for $\rho(\exp(X))f = f$ and $d\rho(X)f = 0$ since $\rho(\exp(X)) = \exp(d\rho(X))$ (a basic result in representation theory, which can be found in [Hall 2013]) where $d\rho$ is the corresponding Lie algebra representation of ρ, which for vector fields is the

Table A.13: HalfCheetah-v2 State and Action Spaces

State Space	X (Unobserved)
	Y
	Orientation Angle
	Rear Hip Angle
	Rear Knee Angle
	Rear Ankle Angle
	Front Hip Angle
	Front Knee Angle
	Front Ankle Angle
	X Velocity
	Y Velocity
	Orientation Angular Velocity
	Rear Hip Angular Velocity
	Rear Knee Angular Velocity
	Rear Ankle Angular Velocity
	Front Hip Angular Velocity
	Front Knee Angular Velocity
	Front Ankle Angular Velocity
Action Space	Rear Hip
	Rear Knee
	Rear Ankle
	Front Hip
	Front Knee
	Front Ankle

Table A.14: Walker2d-v2 State and Action Spaces

State Space	X (Unobserved)
	Y
	Orientation Angle
	Right Hip Angle
	Right Knee Angle
	Right Ankle Angle
	Left Hip Angle
	Left Knee Angle
	Left Ankle Angle
	X Velocity
	Y Velocity
	Orientation Angular Velocity
	Right Hip Angular Velocity
	Right Knee Angular Velocity
	Right Ankle Angular Velocity
	Left Hip Angular Velocity
	Left Knee Angular Velocity
	Left Ankle Angular Velocity
Action Space	Right Hip
	Right Knee
	Right Ankle
	Left Hip
	Left Knee
	Left Ankle

Lie derivative $d\rho(X) = L_X$. Hence carrying over the constraint for each element $\forall X \in \mathfrak{g} : L_X f = 0$ is equivalent to $\forall X \in \mathfrak{g} : \rho(\exp(X))f = f$ which is the same as $\forall g \in G : \rho(g)f = f$. Unpacking the representation ρ_{12} of f, this is just the global equivariance constraint $\forall g \in G : \rho_2(g)^{-1}f(\rho_1(g)x) = f(x)$.

A.3.1.2 Lie Derivative Chain Rule

Suppose we have two functions $h : V_1 \to V_2$ and $f : V_2 \to V_3$, and corresponding representations ρ_1, ρ_2, ρ_3 for the vector spaces V_1, V_2, V_3. Expanding out the definition of ρ_{31},

$$\begin{aligned}\rho_{31}(g)[f \circ h](x) &= \rho_3(g)^{-1} f(h(\rho_1(g)x)) \\ &= \rho_3(g)^{-1} f(\rho_2(g)\rho_2(g)^{-1} h(\rho_1(g)x)) \\ &= \rho_{32}(g)[f] \circ \rho_{21}(g)[h](x).\end{aligned}$$

From the definition of the Lie derivative, and using the chain rule that holds for the derivative with respect to the scalar t, and noting that $g_0 = \mathrm{Id}$ so $\rho(g_0) = \mathrm{Id}$, we have

$$\begin{aligned}\mathcal{L}_X(f \circ h)(x) &= \frac{d}{dt}\Big(\rho_{31}(g_t)[f \circ h](x)\Big)\Big|_0 \\ &= \frac{d}{dt}\Big(\rho_{32}(g_t)[f] \circ \rho_{21}(g_t)[h](x)\Big)\Big|_0 \\ &= \Big(\frac{d}{dt}\rho_{32}(g_t)[f]\big|_{t=0}\Big) \circ \rho_{21}(g_0)[h](x) + \Big[d(\rho_{32}(g_0)[f])\Big|_{h(x)}\Big]\Big(\frac{d}{dt}\rho_{21}(g_t)[h]\big|_{t=0}\Big)(x) \\ &= \Big(\frac{d}{dt}\rho_{32}(g_t)[f]\big|_{t=0}\Big) \circ h(x) + df|_{h(x)}\Big(\frac{d}{dt}\rho_{21}(g_t)[h]\big|_{t=0}\Big)(x) \\ &= (\mathcal{L}_X f) \circ h(x) + df|_{h(x)}(\mathcal{L}_X h)(x),\end{aligned}$$

where $df|_{h(x)}$ is the Jacobian of f at $h(x)$ and $df|_{h(x)}(\mathcal{L}_X h)(x)$ is understood to be the Jacobian vector product of $df|_{h(x)}$ with $(\mathcal{L}_X h)(x)$, equivalent to the directional derivative of f along $(\mathcal{L}_X h)(x)$. Therefore the Lie derivative satisfies a chain rule

A.3.1.3 Stochastic Trace Estimator for Layerwise Metric

Unrolling this chain rule for a sequence of layers $\mathrm{NN}(x) = f_{N:1}(x) := f_N(f_{N-1}(\ldots(f_1(x))))$, or even an autograd DAG, we can identify the contribution that each layer f_i makes to the equivariance error of the whole as the sum of terms $C_i = df_{N:i+1}\mathcal{L}_X f_i$, $\mathcal{L}_X(\mathrm{NN}) = \sum_{i=1}^N C_i$.

Each of these C_i, like $\mathcal{L}_X(\mathrm{NN})$ measure the equivariance error for all of the outputs (which we define to be the softmax probabilities), and are hence vectors of size K where K is the number of classes. In order to summarize the C_i as a single number for plotting, we compute their norm $\|C_i\|$ which satisfy $\|\mathcal{L}_X(\mathrm{NN})\| \leqslant \sum_i \|C_i\|$.

To compute $df_{N:i+1}\mathcal{L}_X f_i$, one can use autograd to perform Jacobian vector products (as opposed to typical vector Jacobian products) and build up $df_{N:i+1}$ in a backwards pass. Unfortunately doing so is quite cumbersome in the PyTorch framework where the large number of available models are implemented and pretrained. A trick which can be used to speed up this computation is to use stochastic trace estimation [Avron and Toledo 2011]. Since vector Jacobian products are cheap and easy, we can compute $\|C_i\|^2 = \mathbb{E}[\hat{A}]$ as the expectation of the estimator $\hat{A} = (1/N)\sum_n^N (z_n^\top C_i)^2 = (1/N)\sum_n^N (z_n^\top df_{N:i+1}\mathcal{L}_X f_i)^2$ with iid. Normal probe vectors $z_n \sim \mathcal{N}(0, I)$, and the quantity $z_n^\top df_{N:i+1}$ which is a standard vector Jacobian product.

One can see that $\mathbb{E}[\hat{A}] = C_i^\top \mathbb{E}[zz^\top] C_i = C_i^\top I C_i = \|C_i\|^2$. We can then measure the variance of this estimator to control for the error and increase N until this error is at an acceptable tolerance (we use $N = 100$ probes). The convergence of this trace estimator is shown in Figure A.3 (right) for several different layers of a ResNet-50. In producing the final layerwise attribution plots, we average the computed quantity $\|C_i\|$ over 20 images from the ImageNet test set.

A.3.2 LEE Theorems

A.3.2.1 LEE and consistency regularization

As shown in Athiwaratkun et al. [2018], consistency regularization with Gaussian input perturbations can be viewed as an estimator for the norm of the Jacobian of the network, but in fact when the perturbations are not Gaussian but from small spatial transformations, consistency regularization actually penalizes the Lie derivative norm. In the Π-model [Laine and Aila 2016] (the most basic form of consistency regularization), the consistency regularization minimizes the norm of the difference of the outputs of the network when two randomly sampled transformations T^a and T^b are applied to the input,

$$\mathrm{L}_{\text{cons}} = \|f(T^a(x)) - f(T^b(x))\|^2. \tag{A.11}$$

Suppose that the two transformations are representations of a given symmetry group and can be written as $T^a = \rho(g_a)$ and $T^b = \rho(g_b)$, and the group elements can be expressed as the flow generated by a linear combination of the vector fields which form the Lie Algebra: $g_a = \Phi_{\sum_i a_i X_i}$ for some coefficients $\{a_i\}_{i=1}^d$ and likewise for g_b. We can define the log map, mapping group elements top their generator values in this basis: $\log(g_a) = a$. Then, assuming a_i are small (and therefore the transformations are small), Taylor expansion yields $\mathrm{L}_{\text{cons}} = \|f(x) + \sum_i a_i \mathcal{L}_{X_i} f(x) + O(a^2) - [f(x) + \sum_j b_j \mathcal{L}_{X_j} f(x) + O(b^2)]\|^2$. Therefore, taking the expectation over the distribution which a and b are sampled over (which is assumed to be centered with $\mathbb{E}[a_i] = \mathbb{E}[b_i] = 0$ as well as the input distribution x, we get that

$$\mathbb{E}_{a,b,x}[\mathrm{L}_{\text{cons}}] = 2\mathbb{E}[\| \sum_i \mathcal{L}_{X_i} f(x)\|_\Sigma^2] + \text{higher order terms}, \tag{A.12}$$

where $\|\|_\Sigma^2$ denotes the norm with respect to the covariance matrix $\Sigma = \mathrm{Cov}(a) = \mathrm{Cov}(b)$.

When the transformations are not parameter space perturbations such as dropout, but input space perturbations like translations (which have been found to be far more important to the overall performance of the method [Athiwaratkun et al. 2018]), we can show that consistency regularization coincides with minimizing the expected Lie derivative norm. In this sense, consistency regularization can be viewed as an intervention for reducing the equivariance error on unlabeled data.

A.3.2.2 Translation LEE and aliasing

Below we show that spatial aliasing directly introduces translation equivariance error as measured by the Lie derivative, where the aliasing operation $A[\cdot]$ is given by Equation 4.2. The Fourier series representation of an image $h(x, y)$ with pixel locations (x, y) is H_{nm} with spatial frequencies (n, m), where the band limited reconstruction

$$h(x,y) = \frac{1}{2\pi}\sum_{nm} H_{nm}e^{2\pi i(xn+ym)} = F^{-1}[H]$$

and F^{-1} is the inverse Fourier transform, and the sums range over frequencies of $-M/2$ to $+M/2$ for both n and m where M is the image height and width (assumed to be square for convenience).

Applying a continuous translation by $t\mathbf{v}$ along vector $\mathbf{v} = (v_x, v_y)$ to the input means resampling the translated band limited continuous reconstruction $h(x, y)$ at the grid points.

$$T_{t\mathbf{v}}[h](x,y) = h(x - tv_x, y - tv_y) = \frac{1}{2\pi}\sum_{n,m=-M/2}^{M/2} H_{nm}e^{2\pi i[(x-tv_x)n+(y-tv_y)m]}$$

To simplify the notation, we will consider translations along only x and suppress the m index of H_{nm}, effectively deriving the result for the translations of a 1d sequence, but that extends straightforwardly to the 2 dimensional case.

$$T_{t\mathbf{v}}[h](x) = h(x - tv_x) = \frac{1}{2\pi}\sum_{n=-M/2}^{M/2} [H_n e^{-2\pi i t v_x n}]e^{2\pi i x n}$$

Applying the aliasing operation, sampling the image to a new size M' (with Nyquist frequency

$M'/2$), we have

$$\begin{aligned}A[T_{t\mathbf{v}}[h]](x) &= \tfrac{1}{2\pi}\sum_{n=-M/2}^{M/2}[H_n e^{-2\pi i t v_x n}]e^{2\pi i x \mathrm{Alias}(n)}\\ &= \tfrac{1}{2\pi}\sum_{n'=-M'/2}^{M'/2}\left[\sum_{n=\mathrm{Alias}^{-1}(n')} H_n e^{-2\pi i t v_x n}\right]e^{2\pi i x n'}\end{aligned}$$

where the last line follows from applying a change of variables $n' = \mathrm{Alias}(n)$.

Applying the final inverse translation (which acts on the M' sampling rate band limited continuous reconstruction), we have

$$T_{-t\mathbf{v}}[A[T_{t\mathbf{v}}[h]]](x) = \tfrac{1}{2\pi}\sum_{n'=-M'/2}^{M'/2}\left[\sum_{n=\mathrm{Alias}^{-1}(n')} H_n e^{-2\pi i t v_x (n-n')}\right]e^{2\pi i x n'}.$$

Taking the derivative with respect to t, we have

$$\begin{aligned}\mathcal{L}_{\mathbf{v}}(A)(h) &= \tfrac{d}{dt}\Big|_0 T_{-t\mathbf{v}}[A[T_{t\mathbf{v}}[h]]]\\ &= \tfrac{1}{2\pi}\sum_{n'=-M'/2}^{M'/2}\left[\sum_{n=\mathrm{Alias}^{-1}(n')} 2\pi i v_x (n'-n) H_n\right]e^{2\pi i x n'}.\end{aligned}$$

Notably, for aliasing when the frequency is reduced by a factor of 2 from downsampling, there are only two values of n that satisfy $\mathrm{Alias}(n) = n'$: the value $n = n'$ and the one that gets aliased down, therefore when multiplied by $n - n'$ the sum $\left[\sum_{n=\mathrm{Alias}^{-1}(n')} 2\pi i v_x (n'-n) H_n\right]$ consists only of a single term.

According to Parseval's theorem, the Fourier transform F is unitary, and therefore the norm of the function as a vector evaluated at the discrete sampling points $x = 1/M', 2/M', \ldots$ is the same as as the norm of the Fourier transform:

$$\|\mathcal{L}_{\text{v}}(A)(h)\|^2 = \|F[\mathcal{L}_{\text{v}}(A)(h)]\|^2$$
$$\|\mathcal{L}_{\text{v}}(A)(h)\|^2 = \sum_{n'=-M'/2}^{M'/2} \left| \sum_{n=\text{Alias}^{-1}(n')} 2\pi i v_x (n'-n) H_n \right|^2$$
$$\|\mathcal{L}_{\text{v}}(A)(h)\|^2 = \sum_{n=-M/2}^{M/2} (2\pi)^2 v_x^2 (\text{Alias}(n)-n)^2 H_n^2,$$

using the fact that only one element is nonzero in the sum. Finally, generalizing to the 2d case, we have

$$\|\mathcal{L}_{\text{v}}(A)(h)\|^2 = (2\pi)^2 \sum_{nm} H_{nm}^2 \big(v_x^2(\text{Alias}(n)-n)^2 + v_y^2(\text{Alias}(m)-m)^2\big), \tag{A.13}$$

showing how the translation Lie derivative norm is determined by the higher frequency components which are aliased down.

A.3.3 Learned Equivariance Experiments

A.3.3.1 Layer-wise Equivariance Baselines

We use EQ-T and EQ-T$_{\text{frac}}$ [Karras et al. 2021] to calculate layer-wise equivariance by caching intermediate representations from the forward pass of the model. For image-shaped intermediate representations, EQ-T samples integer translations in pixels between -12.5% and 12.5% of the image dimensions in pixels. EQ-T$_{\text{frac}}$ is identical but with continuous translation vectors. The individual layer is applied to the transformed input and then the inverse group action is applied to the output, which is compared with the original cached output. Many different normalization could be chosen to compare equivariance errors across layers. The most obvious are $\frac{1}{N}$, $\frac{1}{\sqrt{N}}$, and $\frac{1}{1}$ (no normalization), where $N = C \times H \times W$. As we show in section 4.5, the normalization method can have a large effect of the relative contribution of a layer, despite the decision being relatively arbitrary (in contrast to LEE, which removes the need for doing so as the scale is automatically measured relative to the contribution to the output).

A.3.3.2 Model List

The models included in Figure 1 are

- Early CNNs: ResNets [He et al. 2015], ResNeXts [Xie et al. 2017], VGG [Simonyan and Zisserman 2014], Inception [Szegedy et al. 2016], Xception [Chollet 2017], DenseNet [Huang et al. 2017], MobileNet [Sandler et al. 2018], Blur-Pool Resnets and Densenets [Zhang 2019], ResNeXt-IG [Mahajan et al. 2018a], SeResNe*ts [Hu et al. 2018], ResNet-D [He et al. 2018], Gluon ResNets [Guo et al. 2020; Zhang et al. 2019, 2020], SKResNets [Li et al. 2019], DPNs [Chen et al. 2017]

- Modern CNNs: EfficientNet [Tan and Le 2019a, 2021], ConvMixer [Trockman and Kolter 2022], RegNets [Radosavovic et al. 2020], ResNet-RS, [Bello et al. 2021], ResNets with new training recipes [Wightman et al. 2021], ResNeSts [Zhang et al. 2020], RexNet [Han et al. 2021a], Res2Net [Gao et al. 2019], RepVGG [Ding et al. 2021], NFNets [Brock et al. 2021], XNect [Mehta et al. 2020], MixNets [Tan and Le 2019c], ResNeXts with SSL pretraining [Yalniz et al. 2019], DLA [Yu et al. 2019], CSPNets [Wang et al. 2019], ECA NFNets and ResNets [Brock et al. 2021], HRNet [Sun et al. 2019], MnasNet [Tan et al. 2019]

- Vision transformers: ViT [Dosovitskiy et al. 2020], CoaT [Dai et al. 2021], SwinViT [Liu et al. 2021c], [Bao et al. 2021], CaiT [Touvron et al. 2021c], ConViT [d'Ascoli et al. 2021], CrossViT [Chen et al. 2021], TwinsViT [Chu et al. 2021], TnT [Han et al. 2021b], XCiT [El-Nouby et al. 2021], PiT [Heo et al. 2021], Nested Transformers [Zhang et al. 2022]

- MLP-based architectures: MLPMixer [Touvron et al. 2021b], ResMLP [Touvron et al. 2021a], gMLP [Liu et al. 2021a], MLP-Mixers with (Si)GLU [Wightman 2019]

A.3.3.3 Alternative End-to-End Equivariance Metrics

Discrete Consistency. We adopt the consistency metric from Zhang [2019], which simply measures the fraction of top-1 predictions that match after applying an integer translation to the input (in our case by 10 pixels). Instead of reporting consistency numbers, we report (1 − % matching), so that consistency because a measure of equivariance error. Equivariant models should exhibit end-to-end invariance, high consistency, and low equivariance error.

Expected Group Sample Equivariance. Inspired by work in equivariant architecture design [Finzi et al. 2020; Hutchinson et al. 2021], we provide an additional equivariance metric for comparison against the Lie derivative. Following [Hutchinson et al. 2021], we sample k group elements in the neighborhood of the identity group element, with sampling distribution $\mathcal{D}(G)$, and calculate the sample equivariance error for model f as $\frac{1}{k}\sum_k ||\rho_2^{-1}(g_k)f(\rho_1(g_k)x) - f(x)||$. For translations we take $\mathcal{D}(G)$ to be Uniform$(-5, 5)$ in pixels.

Versus LEE. There are several reasons why the continuous lie derivative metric is preferable over discrete and group sample metrics. Firstly, it allows us to break down the equivariance error layerwise enabling more fine grained analysis in a way not possible with the discrete analog. Secondly, the metric is less dependent on architectural details like the input resolution of the network. For example, for discrete translations by 1 pixel, these translations have a different meaning depending on the resolution of the input, whereas our lie derivatives are defined as the derivative of translations as a fraction of the input size, which is consistently defined regardless of the resolution. Working with the vector space forming the Lie algebra rather than the group also removes some unnecessary freedom in how one constructs the metric. Rather than having to choose an arbitrary distribution over group elements, if we compute the Lie derivatives for a set of basis vectors of the lie algebra, we have completely characterized the space, and all lie derivatives are simply linear combinations of the computed values. Finally, paying attention to continuous transformations reveals the problems caused by aliasing which are far less apparent when considering discrete transformations, and ultimately the relevant transformations are continuous and we should study them directly.

A.3.3.4 LEE for Additional Transformations

Beyond the 3 continuous transformations that we study with Lie derivatives above, there are many more that might reveal important properties of the network. Here we include an three additional transformations–hyperbolic rotation, brightening, and stretch.

Figure A.3 (left) shows that, perhaps surprisingly, models with high accuracy become more equivariant to hyperbolic rotations. We suspect this surprisingly general equivariance to diverse set of continuous transformations is probably the result of improved anti-aliasing learned implicitly by more accurate models. LEE does not identify any significant correlation between brightening or stretch transformations and generalization ability.

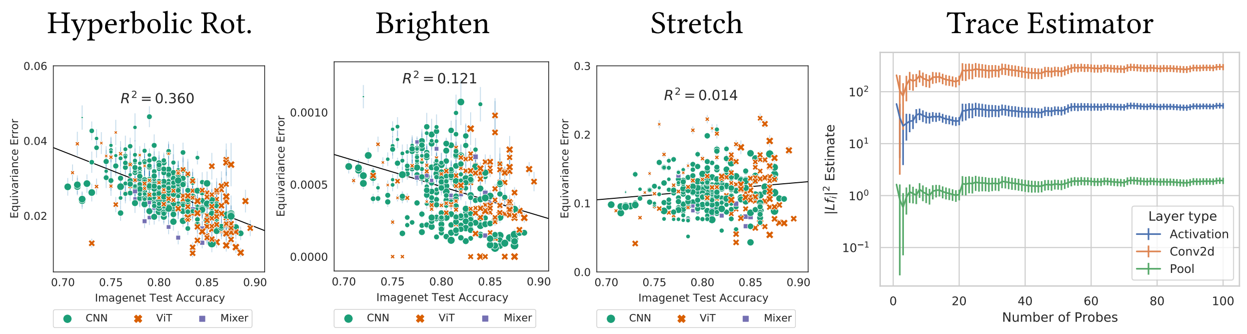


Figure A.3: (Left): Extending Figure 4.5 we show the Lie derivate norm for hyperbolic rotation, brightening, and stretch transformations. We observe that more accurate models are also more equivariant to hyperbolic rotations and to brighten transformation, to a more limited extent. In the case of hyperbolic rotations, this result is surprising, as nothing has directly encouraged this equivariance. One possible explanation is decreased aliasing in models with higher accuracy. Marker size indicates model size. Error bars show one standard error over the images use to evaluate the Lie derivative. **(Right)**: Cumulative mean and standard error of the estimator (computed for translations on a ResNet-50).

A.3.3.5 Rotated MNIST Finetuning

In order to test the ability of SOTA imagenet pre-trained models to learn equivariance competitive with specialized architectures, we adapted the example rotated MNIST notebook available in E2CNN repository [Weiler and Cesa 2019b]. We use the base model and default finetuning procedure from [He et al. 2021], finetuning for 100 epochs, halving the learning rate on loss plateaus.

A.4 Appendix for Understanding the Inductive Biases of Neural Networks

Appendix Outline

The appendix is organized as follows.

- In Appendix A.4.1, we report results for additional bounds for SVHN and ImageNet. We also report the compression size corresponding to our best bound values and compare it to the compression size obtained through standard pruning. Furthermore, in **??** A.4.1.1 we prove why models cannot both be compressible and fit random labels.
- In Appendix A.4.2, we describe how optimization over hyperparameters like the intrinsic dimension impact the PAC-Bayes bound
- In Appendix A.4.3, we show how our PAC-Bayes bound benefit from transfer learning.
- In Appendix A.4.4, we discuss data-dependent priors and their effect on the subspace dimension optimization.
- In Appendix A.4.5, we detail our experimental setup including models, datasets, and hyper-parameter settings for training and bound computation.

- In Appendix A.4.6, we provide a compression perspective to why equivariant models may be more desirable for generalization.
- In Appendix A.4.7, we further discuss how our through our PAC-Bayes compression bounds, we provide evidence that SGD is not necessary for generalization.
- In Appendix A.4.8, we ablate the model size and show how it impacts our bounds and compressibility, we identify the best performing size of models for our bounds.
- In Appendix A.4.9, we present our observations on double descent and their preditability from our PAC-Bayes bounds.
- In Appendix A.4.10, we expand our theoretical discussion and emphasize conceptual differences between our method and previous ones in the literature.
- Lastly, in **??** we provide licensing information on the datasets we use.

A.4.1 Additional Results

In addition to the results reported in Table 5.2, we report the best bounds for SVHN and ImageNet-1k as well as the corresponding compressed size in Tables A.18 and A.19. In Table A.18 we show how compressing the model via intrinsic dimension (ID) yields better results than standard pruning. In this table, we basically run our method but substitute ID with pruning and then proceed by quantizing the remaining weights and encoding them through arithmetic encoding. When pruning we used the standard iterative procedure following Han et al. [2016], for the MNIST model we pruned 98.8% of the weights, for the FMNIST model 97.0% of the weights, for the SVHN model 98.8% of the weights and for both the CIFAR-10 and CIFAR-100 models we pruned 52.1% of the weights and stopped there as the accuracy dropped significantly if we kept pruning.

Error bars on our bounds: We re-run the bounds computation for 10 times and observe that the values are consistent. On average, we obtain ±0.5% variation in our bounds for models trained from scratch and ±0.1% variation for transfer learning models.

A.4.1.1 Models that can fit random labels cannot be compressed

Our ability to construct nonvacuous generalization bounds rests on the ability to construct models which both fit the training data and are highly compressible. However, when the structure in the dataset has been completely destroyed by shuffling the labels, then we do not find that our models are compressible (shown in Figure 5.3 right). This is not just an empirical fact, but one that can be proven apriori: models which fit random labels cannot be compressed. While this result is a trivial consequence of complexity theory, we present an argument here for illustration.

Almost all random datasets are incompressible

When sampling labels uniformly at random, almost all datasets are not substantially compressible. Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$ (where we are only considering the labels y_i, and conditioning on the inputs x_i), and denoting $|\mathcal{D}|$ as the length of the string of labels, the probability that a given dataset can be compressed to size $|\mathcal{D}| - c$ is less than 2^{-c+1}. To see this, one must consider that there are only $\sum_{i=0}^{|\mathcal{D}|-c} 2^i \leqslant 2^{|\mathcal{D}|-c+1}$ programs of length $\leqslant |\mathcal{D}| - c$ (fewer still when restricting to self delimiting programs), and there are $2^{|\mathcal{D}|}$ possible datasets. Therefore averaging over all randomly labeled datasets the fraction which are compressible to less than or equal to $|\mathcal{D}| - c$ bits is at most $2^{|\mathcal{D}|-c+1}/2^{|\mathcal{D}|} = 2^{-c+1}$.

A compressible model which fits the data is a compression of the dataset

Let prior P that includes a specification of the model architecture, and the model h which outputs probabilities for each of the outcomes: $p(y = k \mid x_i) = h(x_i)_k$. We can decompose the (prefix) Kolmogorov complexity of the dataset (given the prior) as

$$K(\mathcal{D} \mid P) \leqslant K(\mathcal{D} \mid h, P) + K(h \mid P). \tag{A.14}$$

The term $K(\mathcal{D} \mid h, P)$ can be interpreted as a model fit term and upper bounded by the total negative log likelihood simply using the model probabilities as a distribution to encode the labels: $K(\mathcal{D} \mid h, P) \leqslant -\sum_i \log_2 h(x_i)_{y_i} + 1 = \mathrm{NLL}(\mathcal{D} \mid h) + 1$.

Using the fact that almost all random datasets are incompressible, and choosing $c = 1 + \log_2(1/\delta)$, we have that with probability at least $1 - \delta$ over all randomly sampled datasets $K(\mathcal{D} \mid P) > |\mathcal{D}| - \log_2(1/\delta) - 1$. Plugging into (A.14) and rearranging, we have with probability $1 - \delta$,

$$K(h|P) \geqslant |\mathcal{D}| - \mathrm{NLL}(\mathcal{D} \mid h) - \log_2(1/\delta) - 2, \tag{A.15}$$

In Figure A.6 we plot the quantity $K(h \mid P) + \mathrm{NLL}(\mathcal{D} \mid h)$ which represents the compressed size of the dataset achieved by our model (related to the minimum description length principle). We see that the value is considerably lower than the size of the dataset $|\mathcal{D}|$, emphasizing that real machine learning datasets such as CIFAR-10 have a very low Kolmogorov complexity and are very unlike those with random labels.

A.4.2 Subspace Dimension Optimization and Hyperparameters in the Universal Prior

The smaller the chosen intrinsic dimension d, the more similar θ is to the initialization θ_0 in (5.3). Consequently, that value of θ is more likely under the universal prior given the shorter description length. Note that in this prior, we condition on the random seed used to generate θ_0 and P. As we optimize over different parameters such as the subspace dimension $d = 1, .., D$, and possibly other hyperparameters such as the learning rate, or number of quantization levels L, we must encode these into our prior and thus pay a penalty for optimizing over them. We can accomodate this very simply by considering the hypothesis h as not just specifying the weights, but also specifying these hyperparameters: $h = (\theta, d, L, \mathrm{lr})$, and therefore using the universal prior $P(h) = 2^{-K(h)}/Z$ we pay additional bits for each of these quantities: $K(h) \leqslant K(\theta \mid d, L) + K(d) + K(L) + K(\mathrm{lr})$. If we optimize over a fixed number H of distinct values known in advance for a given hyperparameter such as L, then we can code L using this information in only $\log_2(H)$ bits. In general, we can also bound the dimensionalities searched over by the maximum D so that $K(d) \leqslant \lceil \log_2 D \rceil$ in any case.

A.4.3 Transfer Learning Bounds

We show the expanded results both with and without transfer learning in Table A.18. When finetuning from ImageNet we use the larger EfficientNet-B0 models rather than the small convnet. Despite the fact that the model is significantly larger than the convnet or resnet models that we use to achieve the best bounds for from scratch training, the difference between the finetuned and pretrained models is highly compressible.

A.4.4 Data Dependent Priors

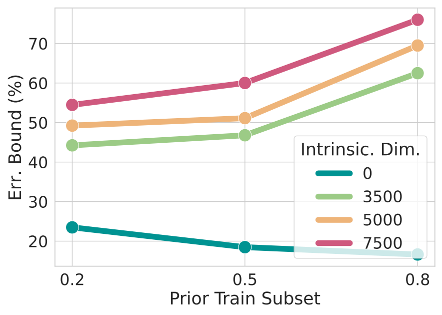
We observe that when using data dependent priors, our optimization over the subspace dimension (and the complexity of the model used to fit the data when measured against the prior) favors very low dimensions and low KL values which we show empirically in Figure A.4. Indeed, a large fraction of the data fitting is moved into fitting a good prior, particularly when the dataset fraction used to train the prior is large. When the prior is already fitted on the data, the final solution can have a very low complexity with respect to that prior without affecting data fit, and is encouraged to do so.

A.4.5 Experimental Details

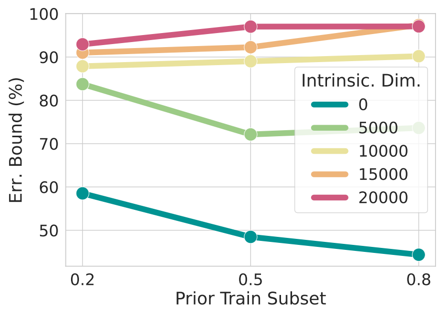
In this section we provide experimental details to reproduce our results.

A.4.5.1 Model Training Details

We use a standard small convolutional architecture for our experiments, which we find produces better bounds than its ResNet counterparts. The architecture is detailed in Table A.20, and we use



(a) CIFAR-10



(b) CIFAR-100

Figure A.4: Data-dependent bounds focus on fitting a good prior. Our bounds using data dependent priors trained using varying fractions of the training dataset. We see that when using data dependent priors, lower intrinsic dimensionalities and lower KL models are favored by the bound.

$k = 16$ for experiments, but this value is ablated in Figure A.6.

Stochastic training: All models were trained for 500 epochs using the Adam optimizer with learning rate 0.001, except for ImageNet which was trained for 80 epochs with SGD using learning rate of 0.05 and weight decay of 0.00002. The model architectures for each dataset are listed below:

- MNIST [LeCun et al. 1998] (+ SVHN [Netzer et al. 2011] Transfer): LeNet-5 [LeCun et al. 1998].
- FashionMNIST [Xiao et al. 2017] (+ CIFAR-10 [Krizhevsky 2009] Transfer): ResNet20 [He et al. 2016b].
- SVHN: ConvNet (Table A.20).
- SVHN + ImageNet Transfer: EfficientNet-B0 [Tan and Le 2019b].
- CIFAR-10: ConvNet.
- CIFAR-10 + ImageNet Transfer: EfficientNet-B0 [Tan and Le 2019b].
- CIFAR-100 [Krizhevsky 2009]: ConvNet.
- CIFAR-100 + ImageNet [Deng et al. 2009] Transfer: EfficientNet-B0.

Full-batch training: We train all models for 3000 epochs, use learning rates equal to 0.1 (MNIST + LeNet-5 and CIFAR-10 + ResNet-18) and 0.5 (CIFAR-10 + ConvNet), and a cosine learning rate scheduler that we warm-up for 10 epochs. We also clip the full gradient to have an L_2-norm of at most 0.25 before performing parameter updates in each epoch [Geiping et al. 2022].

Transfer Learning All previous training details remain the same, except that θ_0 from (5.3) is initialized from a pre-trained checkpoint instead of a random initialization. As typically done in literature, the final classification layer is replaced with a randomly initialized fully-connected layer to account for the number of classes in the downstream task.

A.4.5.2 Bound Hyperparameter Optimization

As explained in Appendix A.4.2, we optimize the bound hyperparameters by considering that the hypothesis of interest h specifies the hyperparameters in addition to the weights. Therefore, we pay bits back for the combination of hyperparemeters that we select. For example, if we are doing a grid search over 2 values of the quantization-aware training learning rate, 2 values of the intrinsic dimensionality values, 2 values of the quantization levels, and use k-means by default, then the number of bits that we pay is $\log_2(2 \times 2 \times 2) = 3$ bits.

Optimizing PAC-Bayes bounds for data-independent priors: Our PAC-Bayesian subspace compression bounds for data-independent priors have 4 hyperparameters that we list here-under alongside the possible values that we consider for each hyperparameter:

- The learning rate for the quantization-aware training, possible values: $\{0.001, 0.003, 0.005, 0.0001\}$.
- The intrinsic dimensionality, possible values: $\{0, 1000, 2500, 3000, 3500, 4000, 5000, 7500, 8000, 10000, 12000, 15000, 20000, 25000, 50000, 100000, 250000, 500000\}$, except for the ImageNet transfer learning which was conducted on the more limited range: $\{500, 1000, 2000, 3000, 4000, 6000, 8000\}$

- The number of quantization levels, possible values: $\{0, 7, 11, 30, 50\}$.
- The quantization initialization, possible values: {uniform, k-means}.

Note that we only use a subset of these hyperparameter values for some datasets, depending on the dataset size and other considerations. For all bound computations, we use arithmetic encoding and 30 epochs of quantization-aware training.

In Table A.21, we summarize the hyperparameters corresponding to the data-independent bounds that we report in Table 5.2.

Optimizing PAC-Bayes bounds for data-dependent priors: In addition to the hyperparameters listed above, we also tune the hyperparameter corresponding to the subset of the training dataset that we use to train the prior on. We consider the following values for the subset of the training dataset: $\{20\%, 50\%, 80\%\}$.

In Table A.22, we summarize the the hyperparameters corresponding to the data-dependent bounds that we report in Table 5.2. The best bounds are obtained for intrinsic dimensionality equal to 0, therefore no quantization is performed.

A.4.5.3 Computational Infrastructure & Resources

Our computational hardware involved a mix of NVIDIA GeForce RTX 2080 Ti (12GB), NVIDIA TITAN RTX (24GB), NVIDIA V100 (32GB), and NVIDIA RTX8000 (48GB). The experiments were managed via W&B [Biewald 2020]. The total computational cost of all experiments (including the ones that do not appear in this work) amounts to ≈ 8000 GPU hours.

A.4.5.4 Breaking Data and Model Structure Experiment

In this experiment we compared our generalization bounds derived for training convolutional networks and MLPs on standard CIFAR10, as well as when data structure is broken by shuffling the pixels or shuffling the labels. We trained for 100 standard epochs with batch size 128 and then another 50 epochs of quantization aware training in all cases. We use 7 quantization levels and uniform quantization initialization for all to simplify. When comparing against an MLP, we use a 3 hidden layer MLP with ReLU nonlinearities, and we feed in the images by flattening them into $3 \times 32 \times 32$ sized vectors. We use 150 hidden units in the intermediate layers of the MLP and choose $k = 46$ in the simple convolutional architecture described in Table A.20 so as to match the parameter count (though slightly smaller models perform slightly better as ablated in Figure A.6).

A.4.6 Equivariance

We conduct a simple experiment to evaluate the extent to which model equivariance has on the compressibility of deep learning models and the tightness of our generalization bounds. We use the rotationally equivariant C_8 WideResNet model from Weiler and Cesa [2019a] which has an 8-fold rotational symmetry, and we also use a non equivariant version of this model. The equivariant model has a depth of 10 and a widen factor of 4 yielding 1.451M parameters. We control for the number of parameters by adjusting the widen factor of the non equivariant model to 4.67 yielding 1.447M parameters.

We evaluate these models both on MNIST and the RotMNIST dataset [Larochelle et al. 2007] consisting of 12K training examples of rotated MNIST digits. As shown in Figure A.5 (a), when paired with the rotationally symmetric RotMNIST dataset, the rotationally equivariant model achieves better bounds and is more compressible than it's non equivariant counterpart despite having the same number of parameters. However, when this symmetry of the dataset is removed by considering standard MNIST, we see that the benefits of equivariance to the generalization bound and compressibility vs the WRN model dissapear.

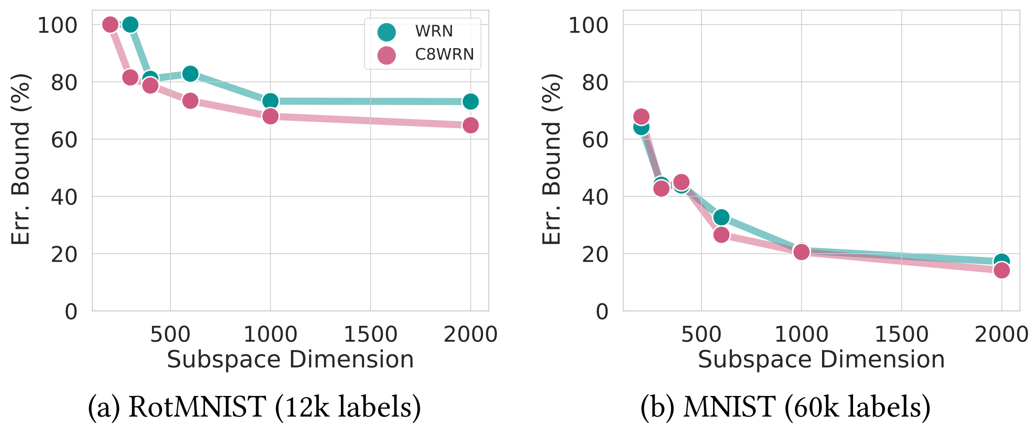


Figure A.5: Rotationally-equivariant models provably generalize better on rotationally-equivariant data. Comparison of rotationally equivariant C_8 WideResNet vs ordinary WideResNet with the same number of parameters on (a) the rotationally equivariant RotMNIST dataset [Larochelle et al. 2007] and (b) the ordinary MNIST dataset. Both models are capable of fitting the data, but the equivariant model yields a more compressible solution when fitting the rotationally equivariant data than the non equivariant model, and hence yields a better generalization bound. (Note the difference in dataset size, RotMNIST has only 12K data points unlike MNIST)

A.4.7 Full-Batch vs. Stochastic Training (SGD)

To further expand on the results that we present in Section 5.6, we study the impact of hyperparameters, namely the weight decay and the architecture choice, on the bounds obtained through full-batch (F-B) training. Table A.23 summarizes these results and we provide the training and bound computation details in Appendix A.4.5. Our PAC-Bayes subspace compression bounds provide similar theoretical guarantees for both full-batch and stochastic training, suggesting that the implicit biases of SGD are not necessary to guarantee good generalization. Moreover, we see that the results are consistent for different configurations, which result in comparable bounds overall.

Transfer learning using full-batch training: We perform full batch training for transfer learning from SVHN to MNIST using LeNet-5 and the same experimental setup described in Appendix A.4.5. Our best PAC-Bayes subspace compression bounds for SVHN to MNIST transfer are 8.7% and 9.0% for full-batch and SGD training, respectively. This finding provides further evidence that good generalization of neural networks, and the success of transfer learning in particular, does not necessarily require stochasticity or additional flatness-inducing procedures to be achieved.

Finally, we note that we optimize over the same set of hyperparameters for the bound computation for both full-batch and stochastic training.

A.4.8 Model Size vs. Compressibility

We perform an ablation to determine how the size of the model affects our generalization bounds. Using the fixed model architecture Table A.20, we vary the width k from 4 to 192. Using our subspace compression scheme, we find that the compression ratio of the model does increase with model size, however the total compressed size still increases slowly making our bounds less strong for larger models. For this paper, we find the sweet spot $k = 16$ is just above the point with equal number of parameters and data points.

We note that this finding leaves room for an improved compression scheme and generalization bounds which are able to explain why even larger models still generalize better. Curiously, when plotting to the total compressed dataset size ($K(h|P) + \text{NLL}$) using the model as a compression scheme, we find that the MDL principle which favors shorter description lengths of the data actually prefers larger models than our PAC-Bayes generalization bounds selects.

A.4.9 Double Descent

Under select conditions, we are able to reproduce the double descent phenomenon in our generalization bounds. In Figure A.7 (right), we show that our bound exhibits a double descent similar to what we see in terms of the test error Figure A.7 (left). The results we show in Figure A.7 (right) are obtained for a fixed intrinsic dimensionality of 35000, but we observed that this middle descent consistently appears in our bounds plots for a given (fixed) intrinsic dimensionality where we

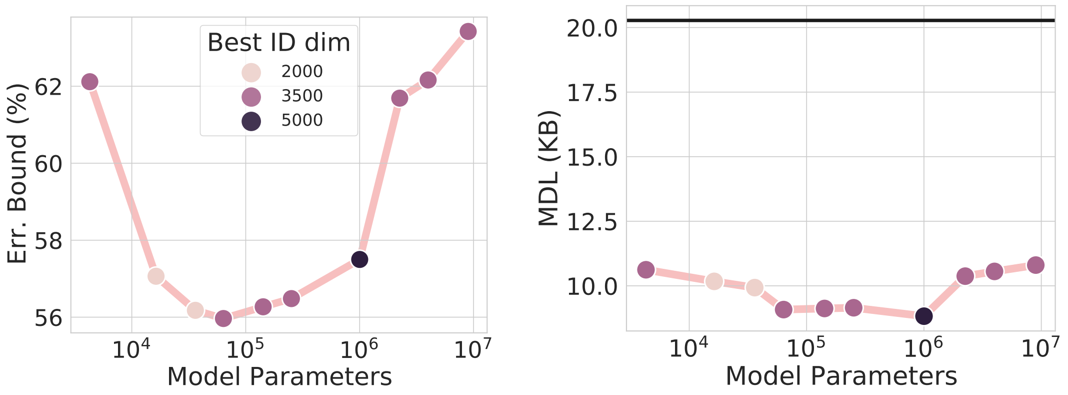


Figure A.6: Model size, compressibility, and MDL. Left: Generalization error bound as a function of model size on the CIFAR10 dataset. The ID subspace dimension that achieves the best bound is shown by the color. In terms of our bound computation, the optimal number of parameters of the network is only slightly above the number of data points. **Right:** The total compressed size ($K(h|P) + \text{NLL}$) of the training dataset using our model as a compression scheme. While the raw labels have size 20.3KB (shown by the black line), the best model compresses the labels down to 8.6KB. Curiously, the compressed dataset size and hence the MDL principle favors larger models than our generalization bounds.

select the best bound for each base width. However, we expect that extending the plot out to larger model widths the bound gets worse again as explained in Appendix A.4.8.

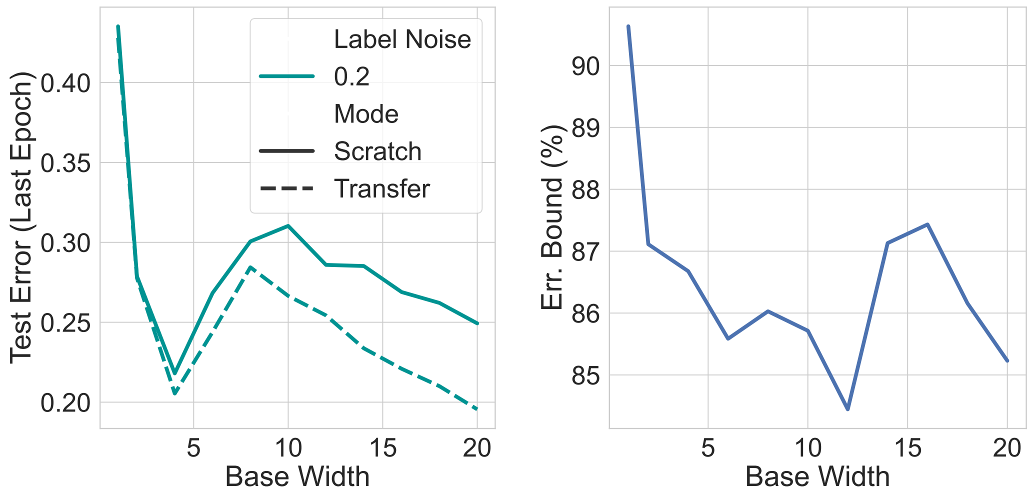


Figure A.7: Our bounds display a double descent as we increase the width. Left: Double descent (in terms of the test error of the last epoch model) observed when varying the width of a ResNets-18 model to fit the CIFAR-10 dataset with label noise equal to 0.2. **Right:** Our bounds showing a similar *double descent* behaviour where the bound starts to worsen only to become better again at a later width. Here we can fix the intrinsic dimensionality to be equal to 35000 and we choose the best subspace compression bound for each base width.

A.4.10 PAC-Bayes Bounds

A.4.10.1 Catoni PAC-Bayes Bound

In our case, since neural networks achieve low training error, we focus on a bound like Catoni [2007] which becomes tighter when $\mathbb{KL}(Q, P)$ is large. This is the same bound used in Zhou et al. [2019].

Theorem A.3 (Catoni [2007]). *Given a* 0-1 *loss* ℓ, *a fixed* $\alpha > 1$ *and a confidence level* $\delta \in (0, 1)$ *then*

$$\underset{\mathbb{E}}{\theta \sim} Q[R(f_\theta)] \leq \inf_{\lambda>1} \Phi^{-1}_{\lambda/N}\left[\underset{\mathbb{E}}{\theta \sim} Q[\hat{R}(f_\theta)] + \frac{\alpha}{\lambda}\left[\mathbb{KL}(Q, P) + \log\frac{1}{\delta} + 2\log\left(\frac{\log(\alpha^2\lambda)}{\log\alpha}\right)\right]\right]$$

holds with probability higher than $1-\delta$ *and where*

$$\Phi^{-1}_{\gamma}(x) = \frac{1-e^{\gamma x}}{1-e^{\gamma}}.$$

A.4.10.2 Variable Length Encoding and Robustness Adjustment

In Zhou et al. [2019], the authors assume a fixed length encoding for the weights. Given that the distribution over quantization levels is highly nonuniform, using a variable length encoding (such as Huffman encoding or arithmetic encoding) can represent the same information using fewer bits. While this choice gives significant benefits, it means that we cannot immediately make use of robustness adjustment from Zhou et al. [2019], where the robustness adjustment comes from considering neighboring models that result from perturbing the weights slightly.

Revisiting the prior derivation in Zhou et al. [2019], we show why the method used for bounding the KL does not transfer over to variable length encodings. In Zhou et al. [2019], the prior used is

$$P = \frac{1}{Z} \sum_{S,Q,C} 2^{-(|S|+|C|+d\lceil \log L \rceil)} \mathcal{N}\left(\hat{w}\left(S, Q, C\right), \tau^2\right)$$

where S denotes the encoding of the position of the pruned weights, C denotes the codebook, Q the codebook value that the weight take, d the number of nonzero weights, L the number of clusters, $\hat{w}$ the quantized weight and τ^2 the prior variance. Note that $\hat{w}$ changes depending on S, Q, C and also note that the fixed-length encoding can be seen in how we sum over $d\lceil \log L \rceil$ options. This prior is a mixture of Gaussians centered at the quantized values. With this choice of prior Zhou et al. [2019] and setting the posterior to be also Gaussian centered at a quantized value, one can upper bound the KL with a computationally tractable term involving the sum over dimensions. Crucially, for their decomposition they use the fact that the size of the encoding $|Q|$ is $d\lceil \log L \rceil$, which is independent of the coding Q and only dependent on the codebook C. Therefore they are able to upper bound the KL.

$$\mathbb{KL}\left(\mathcal{N}\left(\hat{w}, \sigma^2 I_d\right), \sum_{Q} \mathcal{N}(\hat{w}(\hat{S}, Q, \hat{C}), \tau^2)\right) = \sum_{i=1}^{d} \mathbb{KL}\left(\mathcal{N}\left(\hat{w}_i, \sigma^2\right), \sum_{j=1}^{L} \mathcal{N}\left(\hat{w}_j, \tau^2\right)\right)$$

due to the independence of the fixed length encoding to that of the values that each quantized value takes, see appendix in Zhou et al. [2019]. This independence is broken for variable length encoding as the cluster centers and the values that each weight can take are interlinked. Thus, we cannot express and satisfactorily approximate the first high-dimensional KL term as a sum of one-dimensional elements that can be estimated through quadrature or Monte Carlo.

Table A.15: Ant-v2 State and Action Spaces

State Space	X (Unobserved)
	Y (Unobserved)
	Z
	Orientation Quaternion ($4D$)
	Limb 2 Left/Right
	Limb 2 Up/Down
	Limb 3 Left/Right
	Limb 3 Up/Down
	Limb 4 Left/Right
	Limb 4 Up/Down
	Limb 1 Left/Right
	Limb 1 Up/Down
Action Space	Limb 1 Left/Right
	Limb 1 Up/Down
	Limb 2 Left/Right
	Limb 2 Up/Down
	Limb 3 Left/Right
	Limb 3 Up/Down
	Limb 4 Left/Right
	Limb 4 Up/Down

Table A.16: Humanoid-v2 Action Space

Action Space	Torso Forward/Backward
	Torso Z
	Torso Left/Right
	Right Hip Left/Right
	Right Hip Up/Down
	Right Hip Front/Back
	Right Knee Front/Back
	Left Hip Left/Right
	Left Hip Up/Down
	Left Hip Front/Back
	Left Knee Front/Back
	Right Shoulder Left/Right
	Right Shoulder Front/Back
	Right Elbow Front/Back
	Left Shoulder Left/Right
	Left Shoulder Front/Back
	Left Elbow Front/Back

Table A.17: Humanoid-v2 State Space

State Space (Position)	X (Unobserved)
	Y (Unobserved)
	Z
	Orientation Quaternion ($4D$)
	Torso Z
	Torso Forward/Backward
	Torso Left/Right
	Right Hip Left/Right
	Right Knee Left/Right
	Right Hip Up/Down
	Right Knee Up/Down
	Left Hip Left/Right
	Left Knee Left/Right
	Left Hip Up/Down
	Left Knee Up/Down
	Right Shoulder Left/Right
	Right Shoulder Up/Down
	Right Elbow Left/Right
	Left Shoulder Left/Right
	Left Shoulder Up/Down
	Left Elbow Left/Right
State Space (Velocity)	Body Linear Velocity ($3D$)
	Body Angular Velocity ($3D$)
	Torso Z
	Torso Forward/Backward
	Torso Left/Right
	Right Hip Left/Right
	Right Knee Left/Right
	Right Hip Up/Down
	Right Knee Up/Down
	Left Hip Left/Right
	Left Knee Left/Right
	Left Hip Up/Down
	Left Knee Up/Down
	Right Shoulder Left/Right
	Right Shoulder Up/Down
	Right Elbow Left/Right
	Left Shoulder Left/Right
	Left Shoulder Up/Down
	Left Elbow Left/Right

Table A.18: Using our subspace method rather than pruning yields substantially higher compression ratios and hence tighter generalization bounds. We report our error bounds (%) and compressed size ($\mathbb{KL}$ (KB)), 1 KB = 8192 bits. First, we compress the model weights using ID, quantizing its values and then storing them through arithmetic encoding. We then report the bounds obtained by only switching ID to standard pruning. All results are data-independent and obtained with 95% confidence, i.e. $\delta = .05$.

Dataset	ID + Quant + Arith		Pruning + Quant + Arith	
	Err. Bound (%)	$\mathbb{KL}$ (KB)	Err. Bound (%)	$\mathbb{KL}$ (KB)
MNIST	**11.6**	0.4	47.9	6.5
+ SVHN Transfer	**9.0**	0.4		
FashionMNIST	**32.8**	0.8	54.9	3.5
+ CIFAR-10 Transfer	**28.2**	0.9		
SVHN	**36.1**	1.3	74.4	4.3
+ ImageNet Transfer	**29.1**	1.4		
CIFAR-10	**58.2**	1.2	100.0	57.8
+ ImageNet Transfer	**35.1**	1.0		
CIFAR-100	**94.6**	4.1	99.9	50.7
+ ImageNet Transfer	**81.3**	2.8		

Table A.19: Our PAC-Bayesian Subspace Compression Bounds with data-dependent priors compared to state-of-the-art PAC-Bayes non-vacous data-dependent bounds. All results are obtained with 95% confidence, i.e. $\delta = .05$.

Dataset	Err. Bound (%)	SoTA (%)
MNIST	**1.4**	1.5 [Pérez-Ortiz et al. 2021]
FashionMNIST	**10.1**	38 [Dziugaite et al. 2021]
SVHN	**8.7**	–
CIFAR-10	**16.6**	16.7 [Pérez-Ortiz et al. 2021]
CIFAR-100	**44.4**	–
ImageNet	**40.9**	–

Table A.20: Simple convolutional architecture we use to compute our bounds.

ConvNet Architecture
Conv(3,k), BN, ReLU
Conv(k,k), BN, ReLU
Conv(k,2k), BN, ReLU
MaxPool2d(2)
Conv(2k,2k), BN, ReLU
Conv(2k,2k), BN, ReLU
Conv(2k,2k), BN, ReLU
MaxPool2d(2)
Conv(2k,2k), BN, ReLU
Conv(2k,2k), BN, ReLU
Conv(2k,2k), BN, ReLU
GlobalAveragePool2d
Linear(2k,C)

Table A.21: Hyperparameters corresponding to our PAC-Bayesian Subspace Compression Bounds reported in Table 5.2 as well as SVHN and ImageNet to SVHN transfer learning with **data-independent priors**. All bound results are obtained with 95% confidence, i.e. $\delta = .05$.

	Err. Bound (%)	Quant. Learning Rate	Intrinsic Dimensionality	Levels	Quant. Init.
MNIST	**11.6**	0.005	1000	7	Uniform
+ SVHN Transfer	**9.0**	0.005	1000	7	Uniform
FashionMNIST	**32.8**	0.005	2500	7	Uniform
+ CIFAR-10 Transfer	**28.2**	0.005	2500	7	Uniform
SVHN	**36.1**	0.0001	3500	11	Uniform
+ ImageNet Transfer	**29.1**	0.003	4000	7	Uniform
CIFAR-10	**58.2**	0.0001	3500	7	k-Means
+ ImageNet Transfer	**35.1**	0.003	3000	7	Uniform
CIFAR-100	**94.6**	0.0001	10000	11	k-Means
+ ImageNet Transfer	**81.3**	0.003	8000	7	Uniform

Table A.22: Hyperparameters corresponding to our PAC-Bayes bounds reported in Table 5.2 as well as SVHN and ImageNet with **data-dependent priors**. The best bounds are obtained for intrinsic dimensionality equal to 0, therefore no quantization is performed. All bound results are obtained with 95% confidence, i.e. $\delta = .05$.

	Err. Bound (%)	Training Subset (%)
MNIST	**1.4**	50
FashionMNIST	**10.1**	80
SVHN	**8.7**	50
CIFAR-10	**16.6**	80
CIFAR-100	**44.4**	80
ImageNet	**40.9**	50

Table A.23: Our PAC-Bayes subspace compression bounds obtained through full-batch (F-B) training for different configurations and datasets.

Dataset	Architecture	Stochastic Err. Bound (%)	F-B Weight Decay	F-B Err. Bound (%)
MNIST	LeNet-5	11.6	0.01	12.5
			0.001	**11.2**
			0.005	12.0
			0.0001	11.7
CIFAR-10	ResNet-18	74.7	0.01	77.8
			0.001	76.3
			0.005	76.1
			0.0001	**75.3**
CIFAR-10	ConvNet	58.2	0.01	65.8
			0.001	63.6
			0.0001	**61.4**

Bibliography

Abdolhosseini, F., Ling, H. Y., Xie, Z., Peng, X. B., and van de Panne, M. (2019). On learning symmetric locomotion. In *Motion, Interaction and Games*, pages 1–10.

Amos, B., Stanton, S., Yarats, D., and Wilson, A. G. (2020). On the model-based stochastic value gradient for continuous reinforcement learning. *arXiv preprint arXiv:2008.12775*.

Anderson, B., Hy, T. S., and Kondor, R. (2019). Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. (2020). What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*.

Arora, S., Cohen, N., and Hazan, E. (2018a). On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253. PMLR.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018b). Stronger generalization bounds for deep nets via a compression approach. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR.

Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. (2018). There are many consistent explanations of unlabeled data: Why you should average. *arXiv preprint arXiv:1806.05594*.

Avron, H. and Toledo, S. (2011). Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34.

Azulay, A. and Weiss, Y. (2018). Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*.

Bao, H., Dong, L., and Wei, F. (2021). Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. (2022). E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):1–11.

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

Bello, I., Fedus, W., Du, X., Cubuk, E. D., Srinivas, A., Lin, T.-Y., Shlens, J., and Zoph, B. (2021). Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation. *Preprint arXiv:1308.3432v1*.

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

Bietti, A., Venturi, L., and Bruna, J. (2021). On the sample complexity of learning under geometric stability. *Advances in Neural Information Processing Systems*, 34.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1613–1622. JMLR.org.

Bogatskiy, A., Anderson, B., Offermann, J. T., Roussi, M., Miller, D. W., and Kondor, R. (2020). Lorentz group equivariant neural network for particle physics. *arXiv preprint arXiv:2006.04780*.

Bouchacourt, D., Ibrahim, M., and Morcos, A. (2021). Grounding inductive biases in natural images: invariance stems from variations in data. *Advances in Neural Information Processing Systems*, 34:19566–19579.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brock, A., De, S., Smith, S. L., and Simonyan, K. (2021). High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Catoni, O. (2007). PAC-Bayesian Supervised Classification: the Thermodynamics of Statistical Learning. *Institute of Mathematical Statistics Lecture Notes*.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J. T., Sagun, L., and Zecchina, R. (2017). Entropy-sgd: Biasing gradient descent into wide valleys. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chen, C.-F., Fan, Q., and Panda, R. (2021). Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583.

Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., and Feng, J. (2017). Dual path networks.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2018). Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136.

Choi, Y., El-Khamy, M., and Lee, J. (2017). Towards the limit of network quantization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., and Shen, C. (2021). Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765.

Cohen, T., Geiger, M., and Weiler, M. (2018a). A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017*.

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

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. (2018b). Spherical cnns. *arXiv preprint arXiv:1801.10130*.

Cohen, T. S. and Welling, M. (2016b). Steerable cnns. *arXiv preprint arXiv:1612.08498*.

Conrad, K. (2013). Generating sets. *Expository, unpublished paper on the author's personal homepage*.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. *arXiv preprint arXiv:2106.04803*.

d'Ascoli, S., Touvron, H., Leavitt, M., Morcos, A., Biroli, G., and Sagun, L. (2021). Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*.

Dauphin, Y. N., Pascanu, R., Gülçehre, Ç., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2933–2941.

De Sa, C., Re, C., and Olukotun, K. (2015). Global convergence of stochastic gradient descent for some non-convex matrix problems. In *International Conference on Machine Learning*, pages 2332–2341. PMLR.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern*

Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, pages 248–255. IEEE Computer Society.

Ding, N., Chen, X., Levinboim, T., Changpinyo, B., and Soricut, R. (2022). Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. *arXiv preprint arXiv:2203.05126*.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. (2021). Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742.

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

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. (2018). Feature-wise transformations. *Distill*, 3:e11.

Dym, N. and Maron, H. (2020). On the universality of rotation equivariant point cloud networks. *arXiv preprint arXiv:2010.02449*.

Dziugaite, G. K., Hsu, K., Gharbieh, W., Aprino, G., and Roy, D. M. (2021). On the Role of Data in Pac-Bayes Bounds. *The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Dziugaite, G. K. and Roy, D. M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press.

Eberly, W. (2004). Reliable krylov-based algorithms for matrix null space and rank. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 127–134.

El-Nouby, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al. (2021). Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*.

Elesedy, B. (2022). Group symmetry in pac learning. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*.

Elesedy, B. and Zaidi, S. (2021). Provably strict generalisation benefit for equivariant models. *arXiv preprint arXiv:2102.10333*.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2018). A rotation and a translation suffice: Fooling cnns with simple transformations.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2019). Exploring the landscape of spatial robustness. In *International conference on machine learning*, pages 1802–1811. PMLR.

Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. (2017). Polar transformer networks. *arXiv preprint arXiv:1709.01889*.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. (2020). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*.

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

Francis, J. G. (1961). The qr transformation a unitary analogue to the lr transformation—part 1. *The Computer Journal*, 4(3):265–271.

Frey, N., Soklaski, R., Axelrod, S., Samsi, S., Gomez-Bombarelli, R., Coley, C., and Gadepally, V. (2022). Neural scaling of deep chemical models.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. (2020). Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*.

Gao, S., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., and Torr, P. H. (2019). Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*.

Garber, D. and Hazan, E. (2015). Fast and simple pca via convex optimization. *arXiv preprint arXiv:1509.05647*.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8803–8812.

Geiping, J., Goldblum, M., Pope, P. E., Moeller, M., and Goldstein, T. (2022). Stochastic Training Is Not Necessary For Generalization. *The 10th International Conference on Learning Representations (ICLR)*.

Giraud-Carrier, C. and Provost, F. (2005). Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proceedings of the ICML-2005 Workshop on Meta-learning*, pages 12–19.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I. J. and Vinyals, O. (2015). Qualitatively characterizing neural network optimization problems. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15353–15363.

Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., Zhang, A., Zhang, H., Zhang, Z., Zhang, Z., Zheng, S., and Zhu, Y. (2020). Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7.

Guralnick, R. M. (1989). On the number of generators of a finite group. *Archiv der Mathematik*, 53(6):521–523.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018a). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018b). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018c). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hall, B. C. (2013). Lie groups, lie algebras, and representations. In *Quantum Theory for Mathematicians*, pages 333–366. Springer.

Han, D., Yun, S., Heo, B., and Yoo, Y. (2021a). Rethinking channel dimensions for efficient model design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 732–741.

Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., and Wang, Y. (2021b). Transformer in transformer. *arXiv preprint arXiv:2103.00112*.

Han, S., Mao, H., and Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *The 4th International Conference on Learning Representations (ICLR)*.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2021). Masked autoencoders are scalable vision learners. *arXiv:2111.06377*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

He, K., Zhang, X., Ren, S., and Sun, J. (2016c). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2018). Bag of tricks for image classification with convolutional neural networks. *arXiv preprint arXiv:1812.01187*.

Heo, B., Yun, S., Han, D., Chun, S., Choe, J., and Oh, S. J. (2021). Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*.

Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13.

Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural computation*, 9(1):1–42.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. *Advances in neural information processing systems*, 29.

Hume, D. (1978). *A Treatise of Human Nature.* Oxford University Press. revised P.H. Nidditch.

Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. (2021). Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pages 4533–4543. PMLR.

Hutter, M. et al. (2008). Algorithmic complexity.

Ipsen, I. C. (1997). Computing an eigenvector with inverse iteration. *SIAM review*, 39(2):254–291.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. (2021). What are bayesian neural network posteriors really like? In *International conference on machine learning*.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*.

Jiang, J., Dun, C., Huang, T., and Lu, Z. (2018). Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S. (2019). Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Kaggle and EyePacs (2015). Kaggle diabetic retinopathy detection.

Kanavati, F. and Tsuneki, M. (2021). Partial transfusion: on the expressive influence of trainable batch norm parameters for transfer learning. In *Medical Imaging with Deep Learning*, pages 338–353. PMLR.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34.

Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., et al. (2021). Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093.

Kather, J. N., Weis, C.-A., Bianconi, F., Melchers, S. M., Schad, L. R., Gaiser, T., Marx, A., and Z"ollner, F. G. (2016). Multi-class texture analysis in colorectal cancer histology. *Scientific reports*, 6:27988.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kostrikov, I., Yarats, D., and Fergus, R. (2020). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

Lang, L. and Weiler, M. (2020). A wigner-eckart theorem for group equivariant convolution kernels. *arXiv preprint arXiv:2010.10952*.

Langford, J. (2002). *Quantitatively tight sample complexity bounds*. PhD thesis, Carnegie Mellon University.

Langford, J. and Caruana, R. (2001). (not) bounding the true error. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 809–816. MIT Press.

Langford, J. and Seeger, M. (2001). Bounds for Averaging Classifiers. *Tech. rep CMU-CS-01-102, Carnegie Mellon University*.

Laptev, D., Savinov, N., Buhmann, J. M., and Pollefeys, M. (2016). 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.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). 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.

Le, Q. V., Sarlós, T., and Smola, A. (2013). Fastfood: Approximate kernel expansions in loglinear time. *ArXiv*, abs/1408.3060.

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

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

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

Lenc, K. and Vedaldi, A. (2015). Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Li, P., Hastie, T. J., and Church, K. W. (2006). Very sparse random projections. In *KDD '06*.

Li, X., Wang, W., Hu, X., and Yang, J. (2019). Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 510–519.

Li, Z., Malladi, S., and Arora, S. (2021). On the Validity of Modeling SGD with Stochastic Differential Equations (SDEs). *Preprint arXiv:2102.12470.*

Lin, Y., Huang, J., Zimmer, M., Guan, Y., Rojas, J., and Weng, P. (2020). Invariant transform experience replay: Data augmentation for deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6615–6622.

Liu, H., Dai, Z., So, D. R., and Le, Q. V. (2021a). Pay attention to mlps. *arXiv preprint arXiv:2105.08050*.

Liu, I.-J., Yeh, R. A., and Schwing, A. G. (2020). Pic: permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*, pages 590–602. PMLR.

Liu, R., Lehman, J., Molino, P., Such, F. P., Frank, E., Sergeev, A., and Yosinski, J. (2018). An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*.

Liu, Z., Chen, Y., Du, Y., and Tegmark, M. (2021b). Physics-augmented learning: A new paradigm beyond physics-informed learning. *arXiv preprint arXiv:2109.13901*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021c). Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*.

Lyle, C., van der Wilk, M., Kwiatkowska, M., Gal, Y., and Bloem-Reddy, B. (2020). On the benefits of invariance in neural networks. *arXiv preprint arXiv:2005.00178*.

MacKay, D. J. and Mac Kay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and Van Der Maaten, L. (2018a). Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196.

Mahajan, D. K., Girshick, R. B., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. (2018b). Exploring the limits of weakly supervised pretraining. In *ECCV*.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5048–5057.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2018). Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. (2019). On the universality of invariant networks. *arXiv preprint arXiv:1901.09342*.

Maron, H., Litany, O., Chechik, G., and Fetaya, E. (2020). On learning sets of symmetric elements. *arXiv preprint arXiv:2002.08599*.

Martin, V. (2012). Particle physics.

Maurer, A. (2004). A Note on the PAC Bayesian Theorem. *Preprint arXiv 041099v1*.

Mavalankar, A. (2020). Goal-conditioned batch reinforcement learning for rotation invariant locomotion. *arXiv preprint arXiv:2004.08356*.

McAllester, D. A. (1999). PAC-Bayesian Model Averaging. *Proceedings of the 12th Annual Conference on Learning Theory (COLT)*, pages 164–170.

Mehta, D., Sotnychenko, O., Mueller, F., Xu, W., Elgharib, M., Fua, P., Seidel, H.-P., Rhodin, H., Pons-Moll, G., and Theobalt, C. (2020). Xnect. *ACM Transactions on Graphics*, 39(4).

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Nagarajan, V. and Kolter, J. Z. (2019a). Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience. *arXiv preprint arXiv:1905.13344*.

Nagarajan, V. and Kolter, J. Z. (2019b). Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2020). Deep double descent: Where bigger models and more data hurt. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Negrea, J., Dziugaite, G. K., and Roy, D. (2020). In defense of uniform convergence: Generalization via derandomization with an application to interpolating predictors. In *International Conference on Machine Learning*, pages 7263–7272. PMLR.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Neyshabur, B. (2020). Towards learning convolutions from scratch. *arXiv preprint arXiv:2007.13657*.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2018). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Neyshabur, B., Tomioka, R., and Srebro, N. (2014). In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*.

Noether, E. (1971). Invariant variation problems. *Transport theory and statistical physics*, 1(3):186–207.

Olah, C., Cammarata, N., Voss, C., Schubert, L., and Goh, G. (2020). Naturally occurring equivariance in neural networks. *Distill*, 5(12):e00024–004.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951. AAAI Press.

Pérez-Ortiz, M., Rivasplata, O., Shawe-Taylor, J., and Szepersvári, C. (2021). Tighter Risk Certificates for Neural Networks. *Journal of Machine Learning Research 22 (2021) 1-40.*

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. (2021). Do vision transformers see like convolutional neural networks?

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941.*

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

Ravanbakhsh, S., Schneider, J., and Poczos, B. (2017). Equivariance through parameter-sharing. In *International Conference on Machine Learning*, pages 2892–2901. PMLR.

Ravindran, B. and Barto, A. G. (2004). Approximate homomorphisms: A framework for non-exact minimization in markov decision processes.

Ribeiro, A. H. and Schön, T. B. (2021). How convolutional neural networks deal with aliasing. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2755–2759. IEEE.

Rivasplata, O., Tankasali, V. M., and Szepesvári, C. (2019). Pac-bayes with backprop. *arXiv preprint arXiv:1908.07380*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*.

Serviansky, H., Segol, N., Shlomi, J., Cranmer, K., Gross, E., Maron, H., and Lipman, Y. (2020). Set2graph: Learning graphs from sets. *arXiv preprint arXiv:2002.08772*.

Shamir, O. (2015). A stochastic pca and svd algorithm with an exponential convergence rate. In *International Conference on Machine Learning*, pages 144–152. PMLR.

Silver, T., Allen, K., Tenenbaum, J., and Kaelbling, L. (2018). Residual policy learning. *arXiv preprint arXiv:1812.06298*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smidt, T., Geiger, M., and Rackers, J. (2020). github.com/e3nn/e3nn.

Smith, S. L., Elsen, E., and De, S. (2020). On the generalization benefit of noise in stochastic gradient descent. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9058–9067. PMLR.

Solomonoff, R. J. (1964). A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22.

Sukhbaatar, S., Szlam, A., and Fergus, R. (2016). Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*.

Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., and Wang, J. (2019). High-resolution representations for labeling pixels and regions.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile.

Tan, M. and Le, Q. (2019a). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.

Tan, M. and Le, Q. V. (2019b). Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

Tan, M. and Le, Q. V. (2019c). Mixconv: Mixed depthwise convolutional kernels.

Tan, M. and Le, Q. V. (2021). Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*.

Taylor, J., Precup, D., and Panagaden, P. (2008). Bounding performance loss in approximate mdp homomorphisms. *Advances in Neural Information Processing Systems*, 21:1649–1656.

Thiemann, N., Igel, C., Wintenberger, O., and Seldin, Y. (2017). A Strongly Quasiconvex PAC-Bayesian Bound. *28th Annual Conference on Learning Theory (COLT).*

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

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601.*

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., et al. (2021a). Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404.*

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021b). Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR.

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jégou, H. (2021c). Going deeper with image transformers. *arXiv preprint arXiv:2103.17239.*

Trockman, A. and Kolter, J. Z. (2022). Patches are all you need? *arXiv preprint arXiv:2201.09792.*

Turner, W. J. (2006). A block wiedemann rank algorithm. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 332–339.

van der Pol, E., Kipf, T., Oliehoek, F. A., and Welling, M. (2020a). Plannable approximations to mdp homomorphisms: Equivariance under actions. *arXiv preprint arXiv:2002.11963.*

van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., and Welling, M. (2020b). Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33.

van der Wilk, M., Bauer, M., John, S., and Hensman, J. (2018). Learning invariances using the marginal likelihood. In *Advances in Neural Information Processing Systems*, pages 9938–9948.

Vasconcelos, C., Larochelle, H., Dumoulin, V., Romijnders, R., Le Roux, N., and Goroshin, R. (2021). Impact of aliasing on generalization in deep convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10529–10538.

Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., and Welling, M. (2018). Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer.

Wang, C.-Y., Liao, H.-Y. M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., and Hsieh, J.-W. (2019). Cspnet: A new backbone that can enhance learning capability of cnn.

Wang, R., Albooyeh, M., and Ravanbakhsh, S. (2020). Equivariant maps for hierarchical structures. *arXiv preprint arXiv:2006.03627*.

Wang, T. and Ba, J. (2019). Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*.

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

Weiler, M. and Cesa, G. (2019b). General E(2)-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*.

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

Wightman, R. (2019). Pytorch image models. https://github.com/rwightman/pytorch-image-models.

Wightman, R., Touvron, H., and Jégou, H. (2021). Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*.

Wilson, A. G. and Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*.

Winkelmann, J. (2003). Dense random finitely generated subgroups of lie groups. *arXiv preprint math/0309129*.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Wood, J. and Shawe-Taylor, J. (1996). Representation theory and invariant neural networks. *Discrete Applied Mathematics*, 69(1):33–60.

Worrall, D. and Welling, M. (2019). Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems*, pages 7366–7378.

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

Wu, J., Hu, W., Xiong, H., Huan, J., Braverman, V., and Zhu, Z. (2020). On the noisy gradient descent that generalizes as SGD. In *Proceedings of the 37th International Conference on Machine*

Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of *Proceedings of Machine Learning Research*, pages 10367–10376. PMLR.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xiao, T., Dollar, P., Singh, M., Mintun, E., Darrell, T., and Girshick, R. (2021). Early convolutions help transformers see better. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. *CoRR*, abs/1905.00546.

Yin, P., Lyu, J., Zhang, S., Osher, S. J., Qi, Y., and Xin, J. (2019). Understanding straight-through estimator in training activation quantized neural nets. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yu, F., Wang, D., Shelhamer, E., and Darrell, T. (2019). Deep layer aggregation.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

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

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Muller, J., Manmatha, R., Li, M., and Smola, A. (2020). Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*.

Zhang, R. (2019). Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR.

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of freebies for training object detection neural networks. *arXiv preprint arXiv:1902.04103*.

Zhang, Z., Zhang, H., Zhao, L., Chen, T., Arik, S. Ö., and Pfister, T. (2022). Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3417–3425.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. (2019). Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Zhou, Y., Ye, Q., Qiu, Q., and Jiao, J. (2017). Oriented response networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528.

Zhu, S., An, B., and Huang, F. (2021). Understanding the generalization benefit of model invariance from a data perspective. *Advances in Neural Information Processing Systems*, 34.