UNIVERSITY OF CALIFORNIA SAN DIEGO

Symmetry in Neural Networks

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Bo Zhao

Committee in charge:

Professor Rose Yu, Chair
Professor Mikhail Belkin
Professor Sicun Gao
Professor Barna Saha

2026

The Dissertation of Bo Zhao is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2026

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Professor Rose Yu, for providing me with space for exploration and ownership of a research direction. I would also like to thank Professors Misha Belkin, Sicun Gao, and Barna Saha for being on my committee.

I would like to thank the mentors with whom I have been fortunate to work during my PhD. Thank you to Nima Dehmamy for the many hours at the whiteboard, for teaching me physics and new ways of thinking about problems, and for showing me that it is okay to be excited about what I work on. Thank you to Robin Walters for helping me find and shape a research direction, supporting me along the way, and being a role model. Thank you to Iordan Ganev for giving me a glimpse into how mathematicians think and for the influence on my work. Thank you to Robert Gower for being a source of reassurance that we have access to all optimization proofs. Thank you to Hidenori Tanaka for a courageous and exciting research adventure. Thank you to Berkcan Kapusuzoglu, Genta Indra Winata, and Supriyo Chakraborty for the opportunity to explore a new field. Thank you to Eli Grigsby for the help during my job search.

I would also like to thank Eugene Colla and Michael Weissman for giving me a home in physics during my undergraduate years, where I first learned what research is. I am thankful to Le Song for introducing me to computer science research.

I would like to thank my collaborators Peter Eckmann, Salva Rühling Cachay, Tao Wang, Maya Okawa, Eric Bigelow, Ekdeep Singh Lubana, Lucas Laird, Guy Zamir, Aryan Dokania, Luca Zhou, Zehong Wang, Tej Gaonkar, and Edward Berman for generously sharing their knowledge and showing me how talented people work.

I would like to thank my fellow organizers of the weight space learning workshop, the UniReps workshop, and the WiML workshop. I am grateful to be part of these communities.

I would like to thank my friends for their presence, kindness, and care throughout these years. I am also grateful for the gophers, ground squirrels, rabbits, parrots, and owls around my apartment; they brought many moments of joy to my life in San Diego.

Finally, I am grateful for my parents. None of this would have been possible without you.

Chapter 1, in part, is based on the paper published as: Zhao, Bo; Walters, Robin; Yu, Rose. "Symmetry in Neural Network Parameter Spaces." Transactions on Machine Learning Research, issn 2835-8856 (2026). The dissertation author is the primary investigator and author of this paper.

Chapter 2 is a full reprint of material published as: Zhao, Bo; Gower, Robert M.; Walters, Robin; Yu, Rose. "Improving Convergence and Generalization Using Parameter Symmetries." The Twelfth International Conference on Learning Representations (2024). The dissertation author was the primary investigator and author of this paper.

Chapter 3 is a full reprint of material published as: Zhao, Bo; Dehmamy, Nima; Walters, Robin; Yu, Rose. "Understanding Mode Connectivity via Parameter Space Symmetry." In International Conference on Machine Learning, pp. 77441-77460. PMLR 2025. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is based on material published as: Zhao, Bo; Dehmamy, Nima; Walters, Robin; Yu, Rose. "Symmetry teleportation for accelerated optimization." Advances in neural information processing systems 35 (2022): 16679-16690; and Zhao, Bo; Gower, Robert M.; Walters, Robin; Yu, Rose. "Improving Convergence and Generalization Using Parameter Symmetries." The Twelfth International Conference on Learning Representations 2024. The dissertation author was the primary investigator and author of these papers.

Chapter 5, in part, is based on material published as: Zhao, Bo; Gower, Robert M.; Walters, Robin; Yu, Rose. "Improving Convergence and Generalization Using Parameter Symmetries." The Twelfth International Conference on Learning Representations 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in part, is based on the paper from Zhao, Bo; Dehmamy, Nima; Walters, Robin; Yu, Rose. "Finding Symmetry in Neural Network Parameter Spaces." currently being prepared for submission. The dissertation author is the primary investigator and author of this paper.

Chapter 7, in part, is based on the paper published as: Zhao, Bo; Walters, Robin; Yu,

Rose. “Symmetry in Neural Network Parameter Spaces.” Transactions on Machine Learning Research, issn 2835-8856 (2026). The dissertation author is the primary investigator and author of this paper.

VITA

2019	Bachelor of Science in Computer Science, University of Illinois at Urbana-Champaign
2019	Bachelor of Science in Liberal Arts & Sciences in Physics, University of Illinois at Urbana-Champaign
2021	Master of Science in Computer Science, Georgia Institute of Technology
2026	Doctor of Philosophy, University of California San Diego

ABSTRACT OF THE DISSERTATION

Symmetry in Neural Networks

by

Bo Zhao

Doctor of Philosophy in Computer Science

University of California San Diego, 2026

Professor Rose Yu, Chair

In many neural networks, different parameter values can yield the same loss, often due to underlying symmetries in the parameter space. We introduce a general framework for continuous symmetries based on equivariance in activation functions, revealing a new set of nonlinear, data-dependent symmetries. Using these symmetries, we derive topological properties of the minima and identify conserved quantities in gradient flows. As a practical application, we present an algorithm called symmetry teleportation, which leverages parameter symmetries to search loss level sets for points with desired properties. This approach leads to improvements in both convergence and generalization. We also discuss empirical approaches to find symmetries and uncover more structures in neural networks.

Introduction

Modern deep learning models are highly overparameterized, resulting in large sets of parameter configurations that yield the same outputs. A significant portion of this redundancy is explained by symmetries in the parameter space—transformations that leave the network function unchanged. These symmetries shape the loss landscape and constrain learning dynamics, offering a new lens for understanding optimization, generalization, and model complexity that complements existing theory of deep learning.

We propose parameter space symmetry as a valuable, underexplored perspective for understanding neural networks. One of the most important consequences of these symmetries is that they induces nontrivial structure in the level sets of the loss function. In overparameterized networks, where many parameter configurations can yield the same function output, the loss landscape often contains high dimensional manifolds of global minima [31]. Symmetry accounts for much of this degeneracy by mapping parameters within a level set without changing the network's function. Understanding these equivalence classes is essential for analyzing generalization, optimization dynamics, and the loss landscape in deep learning.

While symmetry in the data space has been central to geometric deep learning and equivariant models [26], symmetry in the parameter space has only recently begun to receive sustained attention [162]. Recent work explores parameter symmetry in diverse contexts, including loss landscapes and mode connectivity [52, 37], conserved quantities and training dynamics [82, 134], symmetry-based optimization and model averaging [41, 3], and symmetry-aware sampling in Bayesian inference [144]. This thesis aims to define symmetries in neural network parameter spaces, unify existing theoretical and algorithmic perspectives, and highlight their relevance

to learning theory, optimization, and model analysis. We hope that a clearer understanding of parameter space symmetries will lead to more principled approaches in deep learning theory and practice.

In Section 1, we introduce what symmetry is, why group is a natural structure in studying loss-invariant transformations, and how prevalent symmetries are in the parameter space of neural networks. The rest of this thesis study the effects of symmetries on the loss landscape and learning dynamics, as well as their applications. Section 2 show that conserved quantities associated with linear symmetries can be used to define coordinates along low-loss valleys, linking convergence and generalization with properties at initialization. Section 3 discusses the role of symmetry in the geometry and topology of loss level sets, particularly its influence on the connectedness of minimum. Section 4 and 5 studies an application of symmetry in optimization. Section 6 provides an empirical approach for searching for unknown parameter space symmetries. We conclude with a list of open questions and opportunities in Section 7.

Chapter 1

Motivation and Definitions of Symmetry

In this first chapter, we explain why symmetry is a natural structure to describe neural networks. We then explore various definitions and examples of parameter space symmetry. We start with transformations of the parameters that preserve the feedforward function, and expand to relaxed definitions, which preserve the overall loss function or the feedforward function's value on subsets of data. Along with these definitions, we provide a list of known symmetries and show how they arise from the architecture of neural networks.

1.1 Why Symmetry?

In this section, we show that the set loss-invariant parameter transformations form a group, and therefore easy to describe using the language of symmetry. We provide key concepts in group theory along the way.

Let Param be the space of parameters and $\mathscr{D}$ be the space of data. In supervised learning, $\mathscr{D}$ is decomposed into $\mathscr{D}_{\text{input}} \times \mathscr{D}_{\text{target}}$, where $\mathscr{D}_{\text{input}}$ and $\mathscr{D}_{\text{target}}$ correspond to the space of inputs and target labels, respectively. Let $f\colon \mathsf{Param} \times \mathscr{D}_{\text{input}} \to \mathscr{D}_{\text{target}}$ be a neural network function, and $c\colon \mathscr{D}_{\text{target}} \times \mathscr{D}_{\text{target}} \to \mathbb{R}$ be a function that measures the discrepancy between the neural network's prediction and the ground truth label. The loss function $L\colon \mathsf{Param} \times \mathscr{D} \to \mathbb{R}$ is the composition of f and c. That is, for $(\theta,(x,y)) \in \mathsf{Param} \times \mathscr{D}$, we define the loss $L(\theta,(x,y)) := c\big(f(\theta,x),y\big)$. With occasional exceptions [18, 67], the parameter space of a neural network is typically a real

vector space.

Symmetries are transformations that preserve certain properties of an object. In this paper, we focus on parameter space symmetry, which are transformations of a neural network's parameters that preserve loss. Mathematically, this is the set of bijective transformations $T : \mathsf{Param} \to \mathsf{Param}$ such that $L(w) = L(T(w))$. One may consider restricted sets of symmetry by imposing additional constraints such as linearity or smoothness.

The set of all such transformations forms a group under function composition. To see this, we first examine its structure and properties. Viewing the transformations as functions, we are able to define the composition of two transformations. The composition of two loss-invariant transformations results in another loss-invariant transformation. Since function compositions are associative, the composition of these transformations are associative. Additionally, the identity transformation is always loss preserving, therefore included in our set. Finally, since each transformation is bijective, an inverse exists for every element in the set. These properties of the set of symmetry transformations under composition satisfy the definition of a group.

Definition 1.1.1 (Group)**.** *A group is a set G together with a composition law $\circ$ that satisfies (1) Associativity: $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$ for all g_1, g_2, g_3 in G; (2) Identity: There exists an identity element e in G such that $g \circ e = g$ and $e \circ g = g$ for all g in G; (3) Inverse: For every element g in G, there exists an inverse element g^{-1} such that $g \circ g^{-1} = e$ and $g^{-1} \circ g = e$.*

Following Definition 1.1.1, the set of all loss-invariant and bijective transformations, which we denote by $G_{\Theta,L}$, forms a group. In practice, we typically consider specific subgroups of $G_{\Theta,L}$, since the full group is often difficult to characterize.

For generality and to simplify analysis, we describe symmetry using abstract groups, which focus on the algebraic properties of groups detached from specific transformations. An example that will appear frequently is the $n \times n$ general linear group over $\mathbb{R}$. This group, denoted by $\mathrm{GL}_n(\mathbb{R})$, consists of invertible $n \times n$ real matrices, with composition defined by matrix multiplication. We will also encounter several subgroups of $\mathrm{GL}_n(\mathbb{R})$, including the orthogonal

group $O_n(\mathbb{R})$ which consists of real matrices whose transpose equal inverse, and the positive scaling group $\mathbb{R}^h_{>0}$ which consists of diagonal matrices with positive diagonal entries. Another group relevant to neural network parameter spaces is the symmetric group S_n, which consists of permutations of the set $\{1,2,...,n\}$.

Group actions connect the abstract concept of groups with concrete sets of transformations. A group action is a structure-preserving map from a group into a group of transformations.

Definition 1.1.2 (Group action)**.** *An action of a group G on a set S is a map* $\cdot : G \times S \to S$ *that satisfies* $e \cdot s = s$ *for all* $s \in S$ *and* $g \cdot (g' \cdot s) = (gg') \cdot s$ *for all* g, g' *in G and all s in S.*

A parameter space symmetry can then be described as a group action on the space of parameters that leaves the loss unchanged. In many machine learning settings, these group actions are linear. This leads to the concept of a representation, which maps group elements to invertible matrices and enables the group to act on a vector space by linear transformations.

Definition 1.1.3 (Representation)**.** *A representation of a group G is a homomorphism* $\rho : G \to \mathrm{GL}_n(\mathbb{R})$*, meaning that* $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$ *for all* $g_1, g_2 \in G$*.*

1.2 Functional Neural Network Symmetry

Parameter space symmetry can be defined in various ways, depending on the transformation preserved, i.e. the neural network function or the loss function, and the scope of the data considered, which can range from all possible data, subsets of data, or a particular distribution (Figure 1.3).

This section focuses on the strictest form of parameter space symmetry, which involves transformations that leave the neural network output invariant across all data. Such symmetry preserves the feedforward function. We discuss various general definitions in Section 1.3.

Definition 1.2.1 (Functional neural network symmetry)**.** *Let* Param *be the parameter space of a neural network* $f : \mathsf{Param} \times \mathscr{D}_{input} \to \mathscr{D}_{target}$*. A parameter space symmetry of f is a (possibly*

nonlinear) action of a group G on Param *that leaves f invariant,*

$$f(g \cdot \theta, x) = f(\theta, x), \quad \forall g \in G, \quad \forall \theta \in \text{Param}, \quad \forall x \in \mathscr{D}_{input}.$$

The group G is called a symmetry group of f.

1.2.1 Examples: Symmetries in Common Neural Network Components

Different neural network model architectures give rise to different parameter space symmetries. In the following examples, we examine common components of neural networks and identify specific symmetries associated with each. We begin with a linear network, which offers a clean setting to illustrate how parameter symmetries emerge from rescaling adjacent layers. We work towards realistic examples afterwards.

Example 1.2.2 (Linear)**.** *Consider a two-layer linear neural network $f_{linear}(W_2, W_1, b_2, b_1, X) = W_2(W_1X + b_1) + b_2$, with $(W_2, W_1, b_2, b_1) \in \text{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n} \times \mathbb{R}^m \times \mathbb{R}^h$ and $X \in \mathbb{R}^{n\times k}$. This architecture has a $GL_h(\mathbb{R})$ symmetry, acting on* Param *by*

$$g \cdot (W_2, W_1, b_2, b_1) = (W_2 g^{-1}, gW_1, b_2, gb_1)$$

for $g \in GL_h(\mathbb{R})$ since

$$\begin{aligned} f_{linear}(g \cdot (W_2, W_1, b_2, b_1), X) &= W_2 g^{-1}(gW_1X + gb_1) + b_2 \\ &= W_2(W_1X + b_1) + b_2 \\ &= f_{linear}(W_2, W_1, b_2, b_1, X). \end{aligned}$$

Symmetries of similar forms appear in networks with activation functions, which are more commonly used than linear networks. In particular, many common activation functions are equivariant under a nontrivial group, which leads to following symmetries (Figure 1.1b).

Proposition 1.2.3 ([160]). *Let $\sigma\colon \mathbb{R}^h \to \mathbb{R}^h$ be a function that satisfies $\sigma(gZ) = \rho(g)\sigma(Z)$ for a group G and a representation $\rho\colon G \to \mathrm{GL}_h(\mathbb{R})$. Consider a function $f\colon \mathsf{Param} \times \mathscr{D} \to \mathbb{R}^{m\times n}, (W_2, W_1, b_2, b_1, X) \mapsto W_2\sigma(W_1X + b_1) + b_2$, where $(W_2, W_1, b_2, b_1) \in \mathsf{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n} \times \mathbb{R}^m \times \mathbb{R}^n$ and $X \in \mathbb{R}^{n\times k}$. Then, f admits a functional parameter space symmetry defined by $g \cdot (W_2, W_1, b_2, b_1) \mapsto (W_2\rho(g^{-1}), gW_1, b_2, gb_1)$.*

The symmetries in the next three examples result from equivariance of various pointwise activation functions. A function $\sigma\colon \mathbb{R}^h \to \mathbb{R}^h$ is called pointwise if it is defined as $\sigma(z)_i = \sigma_i(z_i)$ for some scalar functions $(\sigma_i\colon \mathbb{R} \to \mathbb{R})_{i=1}^h$. In practice, σ_i is usually the same across all indices i. A pointwise function $\sigma\colon \mathbb{R}^h \to \mathbb{R}^h$ is homogeneous if there exists a degree $\alpha \in \mathbb{R}^h_{>0}$ such that $\sigma(cz)_i = c^{\alpha_i}\sigma(z)_i$ for all $c \in \mathbb{R}_{>0}$ and $z \in \mathbb{R}^h$. Common homogeneous functions include ReLU, LeakyReLU, and monomials. Since the bias terms are transformed similarly as the other weights, we omit the bias terms in the following examples for brevity.

Example 1.2.4 (Homogeneous Activation). *Consider a two-layer neural network $f(W_2, W_1, X) = W_2\sigma(W_1X)$ with a homogeneous function σ of degree α, where $(W_2, W_1) \in \mathsf{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$ and $X \in \mathbb{R}^{n\times k}$. A symmetry group of this architecture is the positive scaling group $\mathbb{R}^h_{>0}$, which consists of diagonal matrix with positive diagonal entries and act on* Param *by $g \cdot (W_2, W_1) = (W_2g^{-\alpha}, gW_1)$, for $g \in \mathbb{R}^h_{>0}$ [16].*

Example 1.2.5 (Tanh). *Consider a two-layer hyperbolic tangent neural network $f_{tanh}(W_2, W_1, X) = W_2\tanh(W_1X)$, where $(W_2, W_1) \in \mathsf{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$ and $X \in \mathbb{R}^{n\times k}$. This architecture has a sign-flip symmetry, $\mathbb{Z}_2^n$, that consists of diagonal matrix with all diagonal entries in $\{1, -1\}$, acting on* Param *by $g \cdot (W_2, W_1) = (W_2g^{-1}, gW_1)$ [29].*

Example 1.2.6 (Radial neural network [51]). *Often appearing in equivariant neural networks [142, 141], a radial rescaling activation $\sigma\colon \mathbb{R}^h \to \mathbb{R}^h$ has the form $\sigma(z) = f(\|z\|)z$ for some function $f\colon \mathbb{R} \to \mathbb{R}$. A two-layer neural network $f(W_2, W_1, X) = W_2\sigma(W_1X)$ with a radial rescaling activation σ has an $O_h(\mathbb{R})$ symmetry, acting on* Param *by $g \cdot (W_2, W_1) = (W_2g^{-1}, gW_1)$, for $g \in O_h(\mathbb{R})$.*

When σ is G-invariant, which means $\sigma(gZ) = \sigma(Z)$, there exists a group action that acts on only the input weights of σ (Figure 1.1a). We illustrate this in the next two examples.

Example 1.2.7 (Batch Normalization [69])**.** *Batchnorm standardizes inputs of a layer across a mini-batch* $BN(Z)$
$= \frac{Z-E[Z]}{\sqrt{Var[Z]}}$*, where the expectation and variance are computed row-wise over* Z*. Assuming* Z *is a linear feature* $Z = WX$ *where* $W \in \mathsf{Param} = \mathbb{R}^{m\times n}$ *and* $X \in \mathbb{R}^{n\times k}$*, then the batchnorm function has a positive scaling symmetry. Batchnorm is equivariant with respect to the group* $\mathbb{R}^m_{>0}$*, which consists of diagonal matrix with positive diagonal entries, acting on* Param *by* $g \cdot W = gW$*, for* $g \in \mathbb{R}^m_{>0}$ *[82]. Other normalization layers, such as layer normalization [14], group normalization [147], and weight normalization [122], often have scaling symmetry of similar forms.*

Example 1.2.8 (Softmax [25])**.** *The softmax function* $\sigma\colon \mathbb{R}^h \to \mathbb{R}^h$ *is defined as* $\sigma(z)_i = e^{z_i}/\sum_j e^{z_j}$*. Consider the function* $f(W,X) = \sigma(WX)$*, where* $W \in \mathsf{Param} = \mathbb{R}^{m\times n}$ *and* $X \in \mathbb{R}^{n\times k}$*, and* σ *applies on the rows of* WX*. Then* f *has a translation symmetry, acting on* Param *by* $(g \cdot W)_i = W_i + g$ *for each row* W_i*, with* $g \in (\mathbb{R}^n, +)$*, the additive group over* $\mathbb{R}^n$ *[82].*

Many architectures have parameter symmetries given by a symmetric group. This symmetry can often be interpreted as permutation of neurons or components along with their associated weights, as shown in Examples 2.7, 2.8 and Figure 1.1(c). Similar forms of symmetry have also been identified in recurrent neural networks [4].

Example 1.2.9 (Pointwise Activation)**.** *Feedforward networks with pointwise activations that use the same scalar function across coordinates have permutation symmetries. Consider a two-layer neural network* $f(W_2,W_1,X) = W_2\sigma(W_1X)$ *with* $(W_2,W_1) \in \mathsf{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$*,* $X \in \mathbb{R}^{n\times k}$*, and any pointwise activation function* σ *with* $\sigma_i = \sigma_j$ *for all* i, j*. There is a permutation symmetry acting on* Param *by* $g \cdot (W_2,W_1) = (W_2g^{-1}, gW_1)$*, for* $g \in S_h$ *[63].*

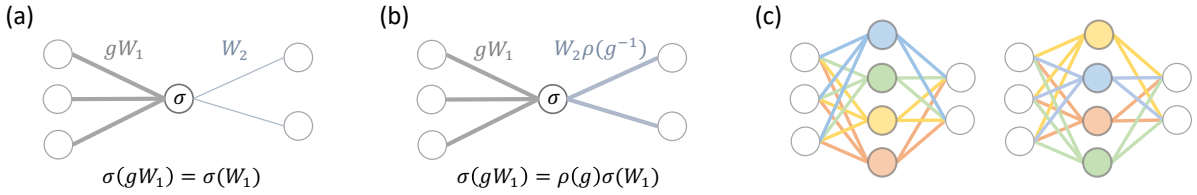


Figure 1.1. Parameter space symmetries arising from neural network architectures. Each circle in the computation graphs represents a neuron or a component in the architecture. Each line segment represents one or a set of parameters. (a) Scaling of the incoming weights of an invariant activation (Examples 1.2.7, 1.2.8). (b) Scaling of the incoming and outgoing weights of an equivariant activation (Examples 1.2.2, 1.2.4-1.2.6). (c) Permutation of neurons (Example 1.2.9) or components (Examples 1.2.10) together with their associated weights.

Example 1.2.10 (Radial basis function networks [27])**.** *In a radial basis function network* $\sum_{i=1}^{k} w_i \varphi\left(\frac{||\mathbf{x}-\mathbf{c_i}||}{b_i}\right)$ *with radial function* $\varphi\colon \mathbb{R}_+ \to \mathbb{R}$ *and parameters* $(w_i, b_i, c_i)_{i=1}^{k} \in \mathsf{Param} = (\mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}^n)^k$*, there is a permutation symmetry acting on* Param *by* $\pi \cdot ((w_i, b_i, c_i)_{i=1}^{k}) = (w_{\pi^{-1}(i)}, b_{\pi^{-1}(i)}, c_{\pi^{-1}(i)})_{i=1}^{k}$*, for* $\pi \in S_k$ *[83].*

Table 1.1. Symmetry in common neural network components.

Name	Architecture	Symmetry Group	Group action
Linear	W_2W_1X	$\mathrm{GL}_h(\mathbb{R})$	$g \cdot (W_2, W_1) = (W_2 g^{-1}, gW_1)$
Homogeneous	$W_2\sigma_{hom}(W_1X)$	$\mathbb{R}^h_{>0}$	$g \cdot (W_2, W_1) = (W_2 g^{-\alpha}, gW_1)$
Tanh	$W_2\sigma_{\tanh}(W_1X)$	$\mathbb{Z}_2^n$	$g \cdot (W_2, W_1) = (W_2 g^{-1}, gW_1)$
Radial rescaling	$W_2\sigma_{radial}(W_1X)$	$O(h)$	$g \cdot (W_2, W_1) = (W_2 g^{-1}, gW_1)$
Batchnorm	$\frac{(WX)_i - \mathrm{E}[(WX)_i]}{\sqrt{\mathrm{Var}[(WX)_i]}}$	$\mathbb{R}^h_{>0}$	$g \cdot W = gW$
Softmax	$\mathrm{softmax}(WX)$	$(\mathbb{R}^n, +)$	$(g \cdot W)_i = W_i + g$
Pointwise	$W_2\sigma_{pointwise}(W_1X)$	S_h	$g \cdot (W_2, W_1) = (W_2 g^{-1}, gW_1)$
Radial basis function	$\sum_{i=1}^{k} w_i \varphi\left(\frac{\|\mathbf{x}-\mathbf{c_i}\|}{b_i}\right)$	S_k	$\pi \cdot (w_i, b_i, c_i) = (w_{\pi^{-1}(i)}, b_{\pi^{-1}(i)}, c_{\pi^{-1}(i)})$

Viewed as a computational graph, a neural network inherits the symmetries of all its subnetworks [158]. Examples 1.2.2–1.2.10 therefore naturally extend to multi-layer networks by applying symmetry to any subset of adjacent layer pairs. Additionally, modern architectures are often constructed from smaller components, many of which have the same form as these

examples. Consequently, they admit the same symmetries, which act on a subspace of their parameter space.

1.2.2 Example: Symmetries in Transformers

To illustrate how symmetries emerge and combine in popular architectures, we delineate the symmetries in transformers by examining their components [155]. Figure 1.2 visualizes the source of these symmetries.

Example 1.2.11 (Attention [140])**.** *The self attention function is a core component of transformers. For key $K \in \mathbb{R}^{m\times p}$, query $Q \in \mathbb{R}^{n\times p}$, and value $V \in \mathbb{R}^{n\times r}$,*

$$Attention(QW^Q, KW^K, VW^V) = softmax\left(\frac{QW^Q(KW^K)^T}{\sqrt{d_k}}\right)VW^V$$

with input embeddings $(Q,K,V) \in \mathbb{R}^{d\times d_m} \times \mathbb{R}^{d\times d_m} \times \mathbb{R}^{d\times d_m}$ and weights $(W_Q, W_K, W_V) \in \mathsf{Param}$ $= \mathbb{R}^{d_m\times d_k} \times \mathbb{R}^{d_m\times d_k} \times \mathbb{R}^{d_m\times d_v}$. This architecture has a $GL_{d_k}(\mathbb{R})$ symmetry, acting on Param *by $g\cdot(W^Q, W^K, W^V) = (W^Q g^{-1}, W^K g^T, W^V)$, for $g \in GL_{d_k}(\mathbb{R})$.*

Building on Example 1.2.11, multi-head attention replicates the attention block across h heads and linearly mixes their outputs, thereby inheriting the $(\mathrm{GL}_{d_k}(\mathbb{R}))^h$ symmetry and introducing additional head-permutation (S_h) and per-head linear symmetries ($(\mathrm{GL}_{d_v}(\mathbb{R}))^h$).

Example 1.2.12 (Multi-head attention [140])**.** *Most transformer architectures use multi-head attentions, which concatenate the output of multiple attentions and apply a linear transformation to the concatenated output. Concretely, a multi-head attention is given by $MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O$, with $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ and additional parameters $W^O \in \mathbb{R}^{hd_v\times d_m}$. The new parameter space is* Param $= (\mathbb{R}^{d_m\times d_k} \times \mathbb{R}^{d_m\times d_k} \times \mathbb{R}^{d_m\times d_v})^h \times \mathbb{R}^{hd_v\times d_m}$*. To simplify notation when describing symmetries, denote $W_i^O \in \mathbb{R}^{d_v\times d_m}$ as the matrix formed by row $(d_v\times i+1)$ to row $(d_v\times(i+1))$ of W^O.*

In addition to the $(\mathrm{GL}_{d_k}(\mathbb{R}))^h$ symmetry inherited from individual attention heads, a multi-head attention admits an S_h and a $(\mathrm{GL}_{d_v}(\mathbb{R}))^h$ symmetry. The S_h symmetry acts as

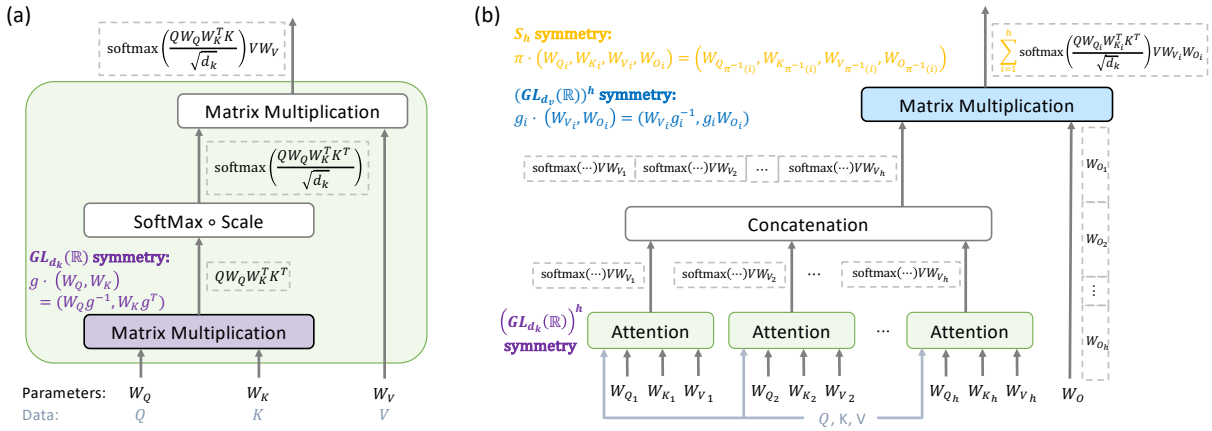


Figure 1.2. Symmetries in (a) the attention mechanism and (b) a multi-headed attention. Dashed rectangles represent output of each layer. Each symmetry is annotated in the same color as that of the component that gives rise to it.

permutation of attention heads: $\pi \cdot (W_i^Q, W_i^K, W_i^V, W_i^O) = (W_{\pi^{-1}(i)}^Q, W_{\pi^{-1}(i)}^K, W_{\pi^{-1}(i)}^V, W_{\pi^{-1}(i)}^O)$*, for* $\pi \in S_h$*. The multiplication of each attention head with* W_O *results in one* $\mathrm{GL}_{d_v}(\mathbb{R})$ *symmetry, acting by* $g_i \cdot (W_i^Q, W_i^K, W_i^V, W_i^O) = (W_i^Q, W_i^K, W_i^V g_i^{-1}, g_i W_i^O)$*, for* $g_i \in \mathrm{GL}_{d_v}(\mathbb{R})$*,* $i = 1, \ldots, h$*.*

Parameters in a transformer can be finetuned after training to adapt to new datasets or tasks. A commonly used method, low-rank adaptation (LoRA) [65], updates pretrained parameters by low-rank matrices to reduce memory and compute. This parametrization adopts a general linear symmetry group.

Example 1.2.13 (LoRA)**.** *In a low-rank adaptation* $W + UV$*, with pretrained weight matrix* $W \in \mathbb{R}^{n \times m}$ *and trainable parameters* $U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{r \times m}$*, there is a* $GL_r(\mathbb{R})$ *symmetry, acting on* (U, V) *by* $g \cdot (U, V) = (Ug^{-1}, gV)$*, for* $g \in GL_r(\mathbb{R})$ *[118].*

1.3 Relaxed Definitions of Symmetry

While functional neural network symmetries ensure invariance of the neural network output across all input data, more general notions of symmetry arise when we relax this requirement. Here, we expand the scope of parameter space symmetry to include transformations that preserve the overall loss function instead of the neural network function, as well as transformations that preserve the output over subsets of data instead of the entire data space.

1.3.1 Loss Symmetry

Loss symmetry refers to parameter transformations that preserve the overall loss but not necessarily the neural network function. Recall that the loss function $L\colon \mathsf{Param} \times \mathscr{D} \to \mathbb{R}$ is often the composition of a model $f\colon \mathsf{Param} \times \mathscr{D}_{\text{input}} \to \mathscr{D}_{\text{target}}$ and a cost function $c\colon \mathscr{D}_{\text{target}} \times \mathscr{D}_{\text{target}} \to \mathbb{R}$. Definition 1.2.1 defines transformations on Param that do not alter the value of f, thereby also preserving L. We now relax this definition to include transformations that are required to preserve L but allowed to change f.

Definition 1.3.1 (Functional loss symmetry)**.** *Let* Param *be the parameter space of a loss function* $L\colon \mathsf{Param} \times \mathscr{D} \to \mathbb{R}$*. A parameter space symmetry of* L *is an action of a group* G *on* Param *that leaves* L *invariant:*

$$L(g \cdot \theta, x) = L(\theta, x), \quad \forall g \in G, \quad \forall \theta \in \mathsf{Param}, \quad \forall x \in \mathscr{D}.$$

Example 1.3.2 (Self-supervised learning with linear network [169])**.** *Consider a self supervised learning setting, with a linear network* $f(W,x) = Wx$ *and a loss* $L(W,x)$ *that depends on* f *only through* $f(x)^T f(x')$ *for data pairs* $x, x' \in \mathbb{R}^n$*. Then,* L *has a rotational symmetry, which acts on* $\mathsf{Param} = \mathbb{R}^{m \times n}$ *by* $g \cdot W = gW$ *for* g *in* $O(m)$*. This differs from the group action on linear networks (Example 1.2.2) since it only acts on one layer and is not canceled by transformations in other layers. Hence, this group action preserves the overall loss* L *but not the feedforward function.*

1.3.2 Data-Dependent Symmetry

In the next definition, we relax the definition of functional symmetries (Definitions 1.2.1 and 1.3.1) by considering parameter transformations that preserve the output for data batches of size n, without guaranteeing invariance on other data. These symmetries depend on the data with respect to which the neural network output is invariant.

Definition 1.3.3 (Data-dependent group action)**.** *A data-dependent group action is a group action of G on $(\mathsf{Param} \times \mathscr{D}^n)$ that acts trivially on $\mathscr{D}^n$. To simplify notation, we drop $\mathscr{D}^n$ from the output and write a data-dependent group action as a map $G \times (\mathsf{Param} \times \mathscr{D}^n) \to \mathsf{Param}$ that satisfies $e \cdot (\theta, X) = \theta$ and $g \cdot (g' \cdot (\theta, X), X) = (gg') \cdot (\theta, X)$ for all g, g' in G, θ in Param, and $X \in \mathscr{D}^n$.*

Definition 1.3.4 (Data-dependent symmetry)**.** *Let Param be the parameter space. Let $F : \mathsf{Param} \times \mathscr{D} \to Y$ be a function with output space Y. A data-dependent symmetry of F is a data-dependent action of a group G on Param that preserves the value of f on a subset of data:*

$$F(g \cdot (\theta, X), x) = F(\theta, x), \quad \forall g \in G,\ \forall \theta \in \mathsf{Param},\ \forall X \in (\mathscr{D}_{input})^n,\ \text{and}\ \ \forall x \in X.$$

Example 1.3.5 (Two-layer network [160])**.** *For a nonzero vector $z \in \mathbb{R}^h$, define a matrix R as*

$$(R_z)_{ij} = \begin{cases} z_i \cos(\alpha_{j-1}) \left(\prod_{k=1}^{j-1} \sin(\alpha_k) \right)^{-1} & \text{if } j \leq i \text{ and } \prod_{k=1}^{i-1} \sin(\alpha_k) \neq 0 \\ -r \sin(\alpha_i) & \text{if } j = i+1 \\ 0 & \text{otherwise.} \end{cases}$$

Consider a two-layer neural network $f(W_2, W_1, X) = W_2 \sigma(W_1 X)$, where $(W_2, W_1) \in \mathsf{Param} = \mathbb{R}^{m \times h} \times \mathbb{R}^{h \times n}$ and $X \in \mathbb{R}^n$. Suppose $\sigma(z)$ is nonzero for any $z \in \mathbb{R}^h$. Then this architecture has a data-dependent $GL_h(\mathbb{R})$ symmetry, which acts on Param by

$$g \cdot (W_2, W_1, x) = (W_2 R_{\sigma(W_1 x)} R^{-1}_{\sigma(g W_1 x)},\ g W_1)$$

and preserves loss value on single data points.

1.3.3 Distribution Symmetry

In practical settings, data can often be viewed as samples from an underlying distribution. The following definition considers parameter transformations that preserve the expected loss

over this distribution. Neural networks related by these transformations are expected to perform similarly on data within the given distribution but not guaranteed to perform similarly on data from different distributions.

Definition 1.3.6 (Distribution symmetry)**.** *Consider a function* $F\colon \mathsf{Param} \times \mathscr{D} \to Y$ *with output space* Y*, where* Param *is the parameter space and data are drawn from a distribution* D*. A distribution symmetry of* F *is an action of a group* G *on* Param *that leaves the expectation of* F *invariant:*

$$\mathbb{E}_{x\sim D}[F(g\cdot\theta,x)] = \mathbb{E}_{x\sim D}[F(\theta,x)], \quad \forall g \in G, \quad \forall \theta \in \mathsf{Param}.$$

Distribution symmetry is often related to averaging effects, where aggregated predictions remain consistent despite variations in the predictions from individual parameters.

Example 1.3.7. *Consider the function* $F\colon \mathbb{R}^{m\times n} \times (\mathbb{R}^n \times \mathbb{R}^n) \to \mathbb{R}^m$ *defined by* $F(W,(x,y)) = Wx - y$*, where* W *is the parameter and* x, y *are data. Assume that* x *is drawn from a distribution* D_x *and for each data pair* (x,y)*,* y *is defined as* $y = W^*x$ *for a fixed matrix* $W^* \in \mathbb{R}^{m\times n}$*. Let* $\bar{x} = \mathbb{E}_{x\sim D_x}[x]$*. Then, the expectation of* F *under the data distribution is:*

$$\mathbb{E}_{x\sim D_x}[F(W,(x,y))] = \mathbb{E}_{x\sim D_x}[Wx - W^*x] = W\bar{x} - W^*\bar{x}.$$

Let G *be the group of all invertible matrices* A *for which* $A\bar{x} = \bar{x}$*. This group acts on the parameter space* $\mathbb{R}^{m\times n}$ *via the action* $A\cdot W = WA^{-1}$*. This action is a distribution symmetry for* F*, because for all* $A \in G$ *and matrix* $W \in \mathbb{R}^{m\times n}$*, we have:*

$$\mathbb{E}_{x\sim D_x}[F(A\cdot W,(x,y))] = \mathbb{E}_{x\sim D_x}[WA^{-1}x - W^*x] = W\bar{x} - W^*\bar{x} = \mathbb{E}_{x\sim D_x}[F(W,(x,y))].$$

The relation among the above definitions are visualized in Figure 1.3. Parameter transformations that preserve the neural network function are guaranteed to preserve the loss function.

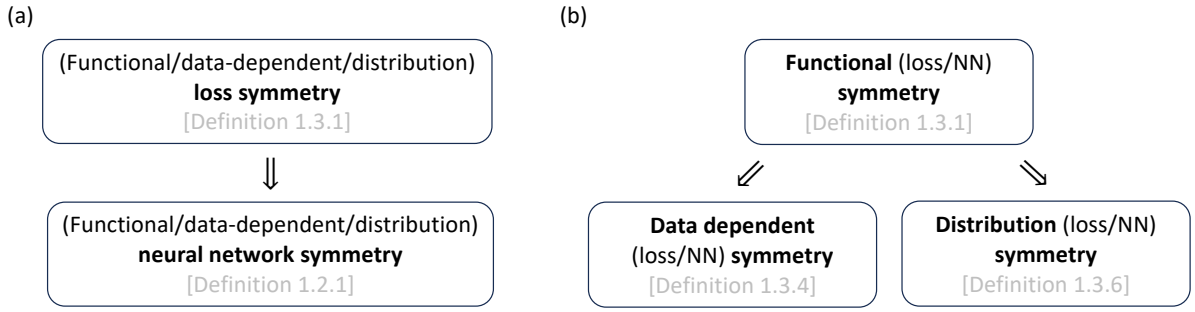


Figure 1.3. Relation among different definitions of parameter space symmetry. Arrows represent the relationship that satisfying one definition implies satisfying the other. (a) Symmetries that preserve the overall loss versus the neural network output. (b) Symmetries defined over the entire input space versus those defined over subsets of data.

Parameter transformations that preserve a function at every data point are guaranteed to preserve the function over every fixed-size batch or distribution. Therefore, neural network symmetry (Definition 1.2.1) implies loss symmetry (Definition 1.3.1). Neural network (or loss) symmetry implies both neural network (or loss) data-dependent symmetry (Definition 1.3.4) and neural network (or loss) distribution loss symmetry (Definition 1.3.6).

Acknowledgments

This chapter, in part, is based on the paper published as: Zhao, Bo; Walters, Robin; Yu, Rose. "Symmetry in Neural Network Parameter Spaces." Transactions on Machine Learning Research, issn 2835-8856 (2026). The dissertation author is the primary investigator and author of this paper.

Chapter 2

Symmetries, Flat Minima, and the Conserved Quantities of Gradient Flow

Empirical studies of the loss landscape of deep networks have revealed that many local minima are connected through low-loss valleys. Yet, little is known about the theoretical origin of such valleys. We present a general framework for finding continuous symmetries in the parameter space, which carve out low-loss valleys. Our framework uses equivariances of the activation functions and can be applied to different layer architectures. To generalize this framework to nonlinear neural networks, we introduce a novel set of nonlinear, data-dependent symmetries. These symmetries can transform a trained model such that it performs similarly on new samples, which allows ensemble building that improves robustness under certain adversarial attacks. We then show that conserved quantities associated with linear symmetries can be used to define coordinates along low-loss valleys. The conserved quantities help reveal that using common initialization methods, gradient flow only explores a small part of the global minimum. By relating conserved quantities to convergence rate and sharpness of the minimum, we provide insights on how initialization impacts convergence and generalizability.

2.1 Introduction

Training deep neural networks (NNs) is a highly non-convex optimization problem. The loss landscape of a NN, which is shaped by the model architecture and the dataset, is generally

very rugged, with the number of local minima growing rapidly with model size [23, 128]. Despite this complexity, recent work has revealed many interesting structures in the loss landscape. For example, NN loss landscapes often contain approximately flat directions along which the loss does not change significantly [48, 52]. Flat minima have been used to build ensemble or mixture models by sampling different parameter configurations that yield similar loss values [52, 21]. However, finding such flat directions is mostly done empirically, with few theoretical results.

One source of flat directions is parameter transformations that keep the loss invariant (i.e. symmetries). Specifically, moving in the parameter space from a minimum in the direction of a symmetry takes us to another minimum. Motivated by the fact that continuous symmetries of the loss result in flat directions in local minima, we derive a general class of such symmetries in this paper.

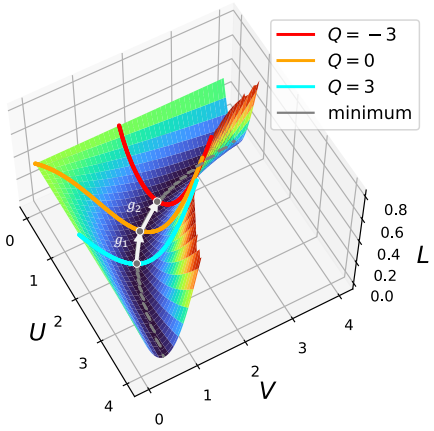


Figure 2.1. Visualization of the extended minimum in a 2-layer linear network with loss $L = \|Y - UVX\|^2$. Points along the minima are related to each other by scaling symmetry $U \to Ug^{-1}$ and $V \to gV$. Conserved quantities, Q, associated with scaling symmetry parametrize points along the minimum.

Our key insight is to focus on *equivariances of the nonlinear activation functions*; most known continuous symmetries can be derived using this framework. Models related by exact equivalence cannot behave differently on different inputs. Hence, for ensembling or robustness tasks, we need to find *data-dependent symmetries*. Indeed, aside from the familiar "linear symmetries" of NN, the framework of equivariance allows us to introduce a novel class of symmetries which act *nonlinearly* on the parameters and are data-dependent. These nonlinear symmetries cover a much larger class of continuous symmetries than their linear counterparts, as they apply for almost any activation function. We provide preliminary experimental evidence that ensembles using these nonlinear symmetries are more robust to adversarial attacks.

Extended flat minima arise frequently in the loss landscape of NNs; we show that symmetry-induced flat minima can be parametrized using *conserved quantities*. Furthermore,

we provide a method of deriving explicit conserved quantities (CQ) for different continuous symmetries of NN parameter spaces. CQ had previously been derived from symmetries for one-parameter groups [82, 134]. Using a similar approach we derive the CQ for general continuous symmetries. This approach *fails* to find CQ for rotational symmetries. Nevertheless, we find the conservation law resulting from the symmetry implies a *cancellation of angular momenta* between layers. To summarize, our contributions are:

1. A general framework based on **equivariance** for finding symmetries in NN loss landscapes.
2. A derivation of the **dimensions of minima** induced by symmetries.
3. A new class of **nonlinear, data-dependent symmetries** of NN parameter spaces.
4. An expansion of prior work on **deriving conserved quantities** (CQ) associated with symmetries, and a discussion of its failure for rotation symmetries.
5. A **cancellation of angular momenta** result for between layers for rotation symmetries.
6. A **parameterization** of symmetry-induced flat minima via the associated CQ.

This paper is organized as follows. First, we review existing literature on flat minima, continuous symmetries of parameter space, and conserved quantities. In Section 2.3, we define continuous symmetries and flat minima, and show how linear symmetries lead to extended minima. We illustrate our constructions through examples of linear symmetries of NN parameter spaces. In Section 2.4, we define nonlinear, data-dependent symmetries. In Section 2.5, we use infinitesimal symmetries to derive conserved quantities for parameter space symmetries, extending the results in [82] to larger groups and more activation functions. Additionally, we show how CQ can be used to define coordinates along flat minima.

In Sections 2.6-2.9, we present a set of experiments aimed at assessing the utility of the nonlinear group action and conserved quantities. We show that the value of conserved quantities

can impact convergence rate and generalizability. We also find the nonlinear action to be viable for ensemble building to improve robustness under certain adversarial attacks.

Exploration of the minimum. While Q is often unbounded, common initialization methods such as [53] limit the values of Q to a small range (Chapter 2.6). As a result, only a small part of the minimum is reachable by the models. Symmetries allow us to explore portions of flat minima that gradient descent rarely reaches.

Convergence rate and generalizability. Conserved quantities are by definition unchanged during gradient flows. By relating the values of conserved quantities to convergence rate and model generalizability, we have access to properties of the trajectory and the final model before the gradient flow starts. This knowledge allows us to choose good conserved quantity values at initialization. In Chapter 2.7, we derive the relation between Q and convergence rate for two example optimization problems, and provide numerical evidence that initializing parameters with certain conserved quantity values accelerates convergence. In Chapter 2.8, we derive the relation between conserved quantities and sharpness of minima in a simple two-layer network, and show empirically that Q values affect the eigenvalues of the Hessian (and possibly generalizability) in larger networks.

Ensemble models. Applying the nonlinear group action allows us to obtain an ensemble without any retraining or searching. We show that even with stochasticity in the data, the loss is approximately unchanged under the group action. The ensemble has the potential to improve robustness under adversarial attacks (Chapter 2.9).

2.2 Related Work

Continuous symmetry in parameter space. Overparametrization in neural networks leads to symmetries in the parameter space [54]. Continuous symmetry has been identified in fully-connected linear networks [135], homogeneous neural networks [16, 38], radial neural networks [51], and softmax and batchnorm functions [82]. We provide a unified framework that

generalizes previous findings, and identify nonlinear group actions that have not been studied before.

Conserved quantities. The imbalance between layers in linear or homogeneous networks is known to be invariant during gradient flow and related to convergence rate [123, 38, 12, 13, 135, 101]. [68] discovered similar conservation laws in natural gradient descents. [82] develop a more general approach for finding conserved quantities for certain one-parameter symmetry groups. [134] relate continuous symmetries to dynamics of conserved quantities using an approach similar to Noether's theorem [112]. We develop a procedure that determines conserved quantities from infinitesimal symmetries, which is closely related to Noether's theorem.

Topology of minimum. The global minimum of overparametrized neural networks are connected spaces instead of isolated points. We show that parameter space symmetries lead to extended flat minima. Previously, [31] proved that the global minima is usually a manifold with dimension equal to the number of parameters subtracted by the number of data points. We derive the dimensionality of the symmetry-induced flat minima and show they are related to the number of infinitesimal symmetry generators and dimension of weight matrices. [128] study permutation symmetry and show that in certain overparametrized networks, the minimum related by permutations are connected. [41] hypothesize that SGD solutions can be permuted to points on the same connected minima. [3] develop algorithms that find such permutations.

2.3 Continuous Symmetries in Deep Learning

In this section, we first summarize our notation for basic neural network constructions (see Appendix A.2 for more details). Then we consider transformations on the parameter space that leave the loss invariant and demonstrate how they lead to extended flat minima.

2.3.1 The Parameter Space and Loss Function

The parameters of a neural network consist of weights[1] $W_i \in \mathbb{R}^{n_i \times m_i}$ for each layer i, where n_i and m_i are the layer output and input dimensions, respectively. For feedforward networks, successive output and input dimensions match: $m_i = n_{i-1}$. We group the widths into a tuple $\mathbf{n} = (n_L, \ldots, n_1, n_0)$, and the parameter space becomes $\mathsf{Param} = \mathbb{R}^{n_L \times n_{i-1}} \times \cdots \times \mathbb{R}^{n_1 \times n_0}$. We denote an element therein as a tuple of matrices $\boldsymbol{\theta} = (W_i \in \mathbb{R}^{n_i \times n_{i-1}})_{i=1}^{L}$. The activation of the i-th layer is a piecewise differentiable function $\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$, which may or may not be pointwise. For $\boldsymbol{\theta} \in \mathsf{Param}$ and input $x \in \mathbb{R}^{n_0}$, the feature vector of the ith layer in feedforward network is $Z_{i+1}(x) = W_{i+1}\sigma(Z_i(x))$, where the juxtaposition '$W\sigma(Z)$' denotes an arbitrary linear operation depending on the context; for example, matrix product, convolution, etc. For simplicity, we largely focus on the case of multilayer perceptrons (MLPs). We denote the final output by $F_{\boldsymbol{\theta}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$, defined as $F_{\boldsymbol{\theta}}(x) = \sigma_L(Z_L(x))$. The "loss function" L of our model is defined as:

$$L : \mathsf{Param} \times \mathbf{Data} \to \mathbb{R}, \qquad L(\boldsymbol{\theta}, (x,y)) = \mathrm{Cost}(y, F_{\boldsymbol{\theta}}(x)). \tag{2.1}$$

where $\mathbf{Data} = \mathbb{R}^{n_0} \times \mathbb{R}^{n_L}$ is the space of data and $\mathrm{Cost} : \mathbb{R}^{n_L} \times \mathbb{R}^{n_L} \to \mathbb{R}$ is a differentiable cost function, such as mean square error or cross-entropy. In the case of multiple samples, we have matrices $X \in \mathbb{R}^{n_0 \times k}$ and $Y \in \mathbb{R}^{n_L \times k}$ whose columns are the k samples[2], and retain the same notation for the feedforward function, namely, $F_{\boldsymbol{\theta}} : \mathbb{R}^{n_0 \times k} \to \mathbb{R}^{n_L \times k}$. Most of our results concern properties of L that hold for any training data. Hence, unless specified otherwise, we take a fixed batch of data $\{(x_i, y_i)\}_{i=1}^{k} \subseteq \mathbf{Data}$, and consider the loss as a function of the parameters only.

Example 2.3.1 (Two-layer network with MSE)**.** *Consider a network with* $\mathbf{n} = (n,h,m)$*, the identity output activation (*$\sigma_L(x) = (x)$*), and no biases. The parameter space is* $\mathsf{Param}(\mathbf{n}) = \mathbb{R}^{n \times h} \times \mathbb{R}^{h \times m}$ *and we denote an element as* $\boldsymbol{\theta} = (U,V)$*. Taking the mean square error cost function, the loss function for data* $(X,Y) \in \mathbb{R}^{n \times k} \times \mathbb{R}^{m \times k}$ *takes the form* $L(\boldsymbol{\theta}, (X,Y)) = \frac{1}{k}\|Y -$

[1] For clarity, we suppress the bias vectors; all results can be extended to include bias; see appendix A.2.

[2] We use capital letters for matrix data and small letters for individual samples.

$U\sigma(VX)\|^2$.

2.3.2 Action of Continuous Groups and Flat Minima

Let G be a group. An action of G on the parameter space Param is a function $\cdot : G \times \mathsf{Param} \to \mathsf{Param}$, written as $g \cdot \boldsymbol{\theta}$, that satisfies the unit and multiplication axioms of the group, meaning $\mathrm{id} \cdot \boldsymbol{\theta} = \boldsymbol{\theta}$ where id is the identity of G, and $g_1 \cdot (g_2 \cdot \boldsymbol{\theta}) = (g_1 g_2) \cdot \boldsymbol{\theta}$ for all $g_1, g_2 \in G$.

Definition 2.3.2 (Parameter space symmetry)**.** *The action $G \times \mathsf{Param} \to \mathsf{Param}$ is a symmetry of L if it leaves the loss function invariant, that is:*

$$L(g \cdot \boldsymbol{\theta}) = L(\boldsymbol{\theta}), \qquad \forall \boldsymbol{\theta} \in \mathsf{Param}, \quad g \in G. \tag{2.2}$$

We describe examples of parameter space symmetries in the next section. Before doing so, we show how a parameter space symmetry leads to flat minima (see Appendix A.2.6):

Proposition 2.3.3. *Suppose $G \times \mathsf{Param} \to \mathsf{Param}$ is a symmetry of $\mathscr{L}$. If $\boldsymbol{\theta}^*$ is a critical point (resp. local minimum) of L, then so is $g \cdot \boldsymbol{\theta}^*$ for any $g \in G$.*

The proof of this result relies on using the differential of the action of g to relate the gradient of L at θ^* with the gradient at $g \cdot \theta^*$. We see that, if $\boldsymbol{\theta}^*$ is a local minimum, then so is every element of the set $\{g \cdot \boldsymbol{\theta}^* \mid g \in G\}$. This set is known as the *orbit* of $\boldsymbol{\theta}^*$ under the action of G. The orbits of different parameter values may be of different dimensions. However, in many cases, there is a "generic" or most common dimension, which is the orbit dimension of any randomly chosen $\boldsymbol{\theta}$.

2.3.3 Equivariance of the activation function

In this section, we describe a large class of linear symmetries of L using an *equivariance* property of the activations between layers. For accessibility, we focus on the example of two

layers with output $F(x) = U\sigma(Vx)$ for $(U,V) \in \mathsf{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$ and $x \in \mathbb{R}^n$. All results generalize to multiple layers by letting $U = W_i$ and $V = W_{i-1}$ be weights of two successive layers in a deep neural network (see Appendix A.2.5). Let $G \subseteq \mathrm{GL}_h(\mathbb{R})$ be a subgroup of the general linear group, and let $\pi : G \to \mathrm{GL}_h(\mathbb{R})$ a representation (the simplest example is $\pi(g) = g$). We consider the following action of the group G on the parameter space Param:

$$g \cdot U = U\pi(g^{-1}), \qquad g \cdot V = gV \tag{2.3}$$

This action becomes a symmetry of L if and only if the following identity holds:

$$\sigma(gz) = \pi(g)\sigma(z) \qquad \forall g \in G, \quad \forall z \in \mathbb{R}^h \tag{2.4}$$

We now turn our attention to examples. To ease notation, we write GL_h instead of $\mathrm{GL}_h(\mathbb{R})$.

Example 2.3.4 (Linear networks)**.** *A simple example of equation 2.4 is that of linear networks, where σ is the identity function: $\sigma(x) = x$. One can take $\pi(g) = g$ and $G = \mathrm{GL}_h$.*

Example 2.3.5 (Homogeneous activations)**.** *Suppose the activation $\sigma : \mathbb{R}^h \to \mathbb{R}^h$ is homogeneous, meaning that (1) σ is applied pointwise in the standard basis and (2) there exists $\alpha > 0$ such that $\sigma(cz) = c^\alpha \sigma(z)$ for all $c \in \mathbb{R}_{>0}$ and $z \in \mathbb{R}^h$. Such an activation is equivariant under the positive scaling group $G \subset \mathrm{GL}_h$ consisting of diagonal matrices with positive diagonal entries. Explicitly, the group G consists of diagonal matrices $g = \mathrm{diag}(\mathbf{c})$ with $\mathbf{c} = (c_1, \ldots, c_h) \in \mathbb{R}^h_{>0}$. For $z = (z_1, \ldots, z_h) \in \mathbb{R}^h$ and $g \in G$, we have $\sigma(gz) = \sum_j \sigma(c_j z_j) = \sum_j c_j^\alpha \sigma(z_j) = g^\alpha \sigma(z)$. Hence, the equivariance equation is satisfied with $\pi(g) = g^\alpha$.*

Example 2.3.6 (LeakyReLU)**.** *This is a special case of homogeneous activation, defined as $\sigma(z) = \max(z,0) + s\min(z,0)$, with $s \in \mathbb{R}_{\geq 0}$. We have $\alpha = 1$, and $\pi(g) = g$.*

Example 2.3.7 (Radial rescaling activations)**.** *A less trivial example of continuous symmetries is the case of a radial rescaling activation [51] where for $z \in \mathbb{R}^h$, we have $\sigma(z) = f(\|z\|)z$ for*

some function $f : \mathbb{R} \to \mathbb{R}$. Radial rescaling activations are equivariant under rotations of the input: for any orthogonal transformation $g \in O(h)$ (that is, $g^T g = I$) we have $\sigma(gz) = g\sigma(z)$ for all $z \in \mathbb{R}^h$. Indeed, $\sigma(gz) = f(\|gz\|)(gz) = g(f(\|z\|)z) = g\sigma(z)$, where we use the fact that $\|gz\| = z^T g^T gz = z^T z = \|z\|$ for $g \in O(h)$. Hence, equation 2.4 is satisfied with $\pi(g) = g$.

We arrive at our first novel result, whose proof appears in Appendix A.2.6.

Theorem 2.3.8. *The dimension of a generic orbit in* Param *under the appropriate symmetry group is given as follows. The cases are divided based on whether $h \leq \max(n,m)$ or not.*

		Orbit Dimension	
Activation	Symmetry Group	$h \leq \max(n,m)$	$h \geq \max(n,m)$
Identity	$\mathrm{GL}_h(\mathbb{R})$	h^2	$h(n+m) - nm$
Homogeneous	Positive rescaling	h	$\max(n,m)$
Radial rescaling	$O(h)$	$\binom{h}{2}$	$\binom{h}{2} - \binom{h-\max(m,n)}{2}$

As an aside, we note that a familiar example where equation 2.4 is satisfied involves the permutation of neurons. More precisely, suppose σ is pointwise and let G be the finite group of $h \times h$ permutation matrices. Then equation 2.4 holds with $\pi(g) = g$. However, the permutation group is finite (0-dimensional), and so does not imply the presence of flat minima.

2.3.4 Infinitesimal Symmetries

Deriving conserved quantities from symmetries requires the infinitesimal versions of parameter space symmetries. Recall that any smooth action of a matrix Lie group $G \subseteq \mathrm{GL}_h$ induces an action of the infinitesimal generators of the group, i.e., elements of its Lie algebra. Concretely, let $\mathfrak{g} = \mathrm{Lie}(G) = T_I G$ be the Lie algebra, which can be identified with a certain subspace of matrices in $\mathfrak{gl}_h = \mathbb{R}^{h \times h}$. For every $M \in \mathfrak{g}$, we have an exponential map $\exp_M : \mathbb{R} \to G$ defined as $\exp_M(t) = \sum_{k=0}^{\infty} \frac{(tM)^k}{k!}$. If $\rho : G \to \mathrm{GL}_h$ is a (linear) representation, then the *infinitesimal action* is given by $d\rho : \mathfrak{g} \to \mathfrak{gl}_h$ by $d\rho(M) = \frac{d}{dt}\big|_0 \rho(\exp_M(t))$. In the case of the

action appearing in equation 2.3, the corresponding infinitesimal action of the Lie algebra $\mathfrak{g}$ induced by equation 2.3 is given by:

$$M \cdot U = -U d\pi(M), \qquad M \cdot V = MV \tag{2.5}$$

More generally, suppose G acts linearly on parameter space (see Appendix A.2 for non-linear versions). Set d to be the dimension of the parameter space[3], and make the identification $\mathsf{Param} \simeq \mathbb{R}^d$ by flattening matrices into column vectors. The general linear group $\mathrm{GL}(\mathsf{Param}) \simeq \mathrm{GL}_d(\mathbb{R})$ consists of all invertible linear transformations of Param. Suppose G is a subgroup of $\mathrm{GL}(\mathsf{Param})$, so its Lie algebra $\mathfrak{g}$ is a Lie subalgebra of $\mathfrak{gl}_d = \mathbb{R}^{d\times d}$. For $M \in \mathfrak{g}$ and $\boldsymbol{\theta} \in \mathsf{Param}$, the *infinitesimal action* is given simply by matrix multiplication: $M \cdot \boldsymbol{\theta}$.

In the case of a parameter space symmetry, the invariance of L translates into the following orthogonality condition, where the inner product $\langle , \rangle : \mathsf{Param} \times \mathsf{Param} \to \mathbb{R}$ is calculated by contracting all indices, e.g. $\langle A, B \rangle = \sum_{ijk\ldots} A_{ijk\ldots} B_{ijk\ldots}$.

Proposition 2.3.9. *Let G be a matrix Lie group and a symmetry of L. Then the gradient vector field is point-wise orthogonal to the action of any $M \in \mathfrak{g}$:*

$$\langle \nabla_{\boldsymbol{\theta}} L \, , \, M \cdot \boldsymbol{\theta} \rangle = 0, \qquad \forall \boldsymbol{\theta} \in \mathsf{Param} \tag{2.6}$$

2.4 Nonlinear Data-Dependent Symmetries

For common activation functions, the equivariance $\sigma(gz) = \pi(g)\sigma(z)$ of equation 2.4 holds only for g belonging to a relatively small subgroup of GL_h. For ReLU, g must be in the positive scaling group, while for the usual sigmoid activation, the equation only holds for trivial $g = \mathrm{id}$. However, under certain conditions, it is possible to define a nonlinear action of the *full* GL_h which applies to many different activations. The subtlety of such an action is that it is data-dependent, which means that, for any $g \in \mathrm{GL}_h$, the transformation of the parameter space

[3] In terms of the widths, we have $d = \sum_{i=1}^{L} n_i n_{i-1}$.

depends on the input data[4] x.

The nonlinear action.

For any nonzero vector $z \in \mathbb{R}^h$, let $(r, \alpha_1, \ldots, \alpha_{h-1})$ be the spherical coordinates[5] of z, and define the following h by h matrix:

$$(R_z)_{ij} = \begin{cases} z_i \cos(\alpha_{j-1}) \left(\prod_{k=1}^{j-1} \sin(\alpha_k) \right)^{-1} & \text{if } j \leq i \text{ and } \prod_{k=1}^{i-1} \sin(\alpha_k) \neq 0 \\ -r \sin(\alpha_i) & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_0 = 0$ by convention. We observe that R_z is the product of a rotation matrix and rescaling by $|z|$. Moreover, since $z \neq 0$, the first column of R_z is the unit vector $z/|z|$ and R_z has inverse given by $R_z^{-1} = \frac{1}{|z|^2} R_z^T$. Using these facts, one arrives at the following result, stated in the case of a two-layer neural network with notation from Section 2.3.3, and proven in Appendix A.3:

Theorem 2.4.1. *Suppose $\sigma(z)$ is nonzero for any $z \in \mathbb{R}^h$. Then there is an action $\mathrm{GL}_h \times (\mathsf{Param} \times \mathbb{R}^n) \to \mathsf{Param} \times \mathbb{R}^n$ given by*

$$g \cdot (U, V, x) = (U R_{\sigma(Vx)} R_{\sigma(gVx)}^{-1} \ , \ gV \ , \ x). \tag{2.7}$$

The evaluation of the feedforward function at x is unchanged: $F_{(U,V)}(x) = F_{(UR_{\sigma(Vx)}R_{\sigma(gVx)}^{-1}, gV)}(x)$.

We emphasize that a necessary and sufficient condition for the particular action of Theorem 2.4.1 to be well-defined is that $\sigma(z)$ be nonzero for any $z \in \mathbb{R}^h$; this is the case for usual sigmoid. Moreover, in Appendix A.3.2, we provide a generalization to the case where $\sigma(z)$ is only required to be nonzero for any *nonzero* $z \in \mathbb{R}^h$, a condition satisfied by hyperbolic tangent, leaky ReLU, and many other activations. The cost of such a generalization is a restriction

[4] That is, rather than being a map $\mathrm{GL}_h \times \mathsf{Param} \to \mathsf{Param}$ satisfying the group action axioms, a data-dependent action is a map $\mathrm{GL}_h \times (\mathsf{Param} \times \mathbb{R}^n) \to \mathsf{Param} \times \mathbb{R}^n$ satisfying the same axioms.

[5] Hence, $r = |z|$ is the norm, and the i-th coordinate of z is $z_i = r \cos(\alpha_i) \prod_{k=1}^{i-1} \sin(\alpha_k)$, where $\alpha_h = 0$.

to a 'non-degenerate locus' of $\mathsf{Param} \times \mathbb{R}^n$ where $Vx \neq 0$. Theorem 2.4.1 also generalizes to mutli-layer networks, as explained in Appendix A.3.3. We have the following explicit algorithm to compute the action of Theorem 2.4.1:

0. Input: weight matrices (U,V), input vector $x \in \mathbb{R}^n$, matrix $g \in \mathrm{GL}_h$.

1. Determine the spherical coordinates of $\sigma(Vx)$ and $\sigma(gVx)$, and construct the matrices $R_{\sigma(Vx)}$ and $R_{\sigma(gVx)}$.

2. Compute the inverse $R^{-1}_{\sigma(gVx)} = \frac{1}{|\sigma(gVx)|^2} R^T_{\sigma(gVx)}$.

3. Set $U' = UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)}$ and $V' = gV$.

4. Output: the transformed weights (U',V'). The data $x \in \mathbb{R}^n$ remains unchanged.

Lipschitz bounds.

Unlike the exact symmetries of Section 2.3, a data-dependent action may alter the loss in the *function space*. This is evident from equation 2.7: while the transformed and original feedforward functions have the same value at x, they will differ at other points. That is, if $\tilde{x} \in \mathbb{R}^h$ is an input value different from x, then $F_{(U,V)}(\tilde{x}) \neq F_{(UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)}, gV)}(\tilde{x})$ in general.

However, the transformed feedforward function will differ from the original one in a controlled way. More precisely, when σ is Lipschitz continuous, we show that there is a bound on how much the Lipschitz bound of the feedforward changes after the nonlinear action. The relevance of such a bound originates in the fact that we expect the distance between data points to encode important information about shared features. To be more specific, fix weight matrices (U,V), which provide the feedforward function $F(\tilde{x}) = U\sigma(V\tilde{x})$. For any input vector $x \in \mathbb{R}^n$ and matrix $g \in \mathrm{GL}_h$, the transformed weight matrices $(UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)}, gV)$ provide a new feedforward function given by:

$$F^{(g,x)}_{(U,V)} : \mathbb{R}^n \to \mathbb{R}^m \qquad F^{(g,x)}_{(U,V)}(\tilde{x}) = UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)}\sigma(gV\tilde{x}) \tag{2.8}$$

Proposition 2.4.2 (Lipschitz bounds from equivariance)**.** *Let σ be Lipschitz continuous with Lipschitz constant η. Then $F^{(g,x)}_{(U,V)}$ is Lipschitz continuous with bound $\eta\|U\|\|V\|\frac{|\sigma(Vx)|\|g\|}{|\sigma(gVx)|}$.*

In particular, the Lipschitz bound of the original feedforward function is $\eta\|U\|\|V\|$. Thus, if it happens that $|\sigma(Vx)|\|g\| < |\sigma(gVx)|$, then the Lipschitz bound decreases when transforming the parameters. Additionally, we observe that the nonlinear action does not disrupt latent distribution of data significantly. See Appendix A.3.5 for proof of Proposition 2.4.2, which relies on iterative applications of the Cauchy-Schwarz inequality, as well as the fact that $\|R_z^{\pm 1}\| = |z|^{\pm 1}$.

General equivariance.

The action described is an instance of a more general framework of equivariance. Specifically, a map $c : \mathrm{GL}_h \times \mathbb{R}^h \to \mathrm{GL}_h$ is said to be an *equivariance* if it satisfies (1) $c(\mathrm{id}_h, z) = \mathrm{id}_h$ for all z, and (2) $c(g_1, g_2 z)c(g_2, z) = c(g_1 g_2, z)$ for all $g_1, g_2 \in \mathrm{GL}_h$ and z. These two conditions on c translate directly into the unit and multiplication axioms of a group[6], generalizing $\pi(g_1 g_2) = \pi(g_1)\pi(g_2)$, and $\pi(\mathrm{id}_h) = \mathrm{id}_h$. Every equivariance gives rise to a nonlinear action of GL_h on $\mathsf{Param} \times \mathbb{R}^h$ given by $g \cdot (U,V,x) = (Uc(g,Vx)^{-1}, gV, x)$. This action is a symmetry preserving the loss if and only if the following generalization of equation 2.4 holds:

$$\text{General Equivariance:} \qquad \sigma(gz) = c(g,z)\sigma(z) \qquad \forall g \in \mathrm{GL}_h \; \forall z \in \mathbb{R}^h \tag{2.9}$$

An explicit example of such an equivariance is $c(g,z) = R_{\sigma(gz)} R^{-1}_{\sigma(z)}$, and Proposition 2.4.2 generalizes to any general equivariance by replacing $\frac{|\sigma(Vx)|\|g\|}{|\sigma(gVx)|}$ with $\|c(g,Vx)^{-1}\|$.

2.5 Conserved Quantities of Gradient Flow

We have shown that continuous symmetries lead us along extended flat minima in the loss landscape. In this section, we identify quantities that (partially) parameterize these minima.

[6] In fact, c defines a GL_h-equivariant structure on the tangent bundle of $\mathbb{R}^h$.

We first show that certain real-valued functions on the parameter space remain constant during gradient flow. We refer to such functions as *conserved quantities*. Applying symmetries changes the value of the conserved quantity. Therefore, conserved quantities can be used to parameterize flat minima.

Gradient flow (GF).

Recall that GD proceeds in discrete steps with the update rule $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \varepsilon \nabla L(\boldsymbol{\theta}_t)$ where ε is the learning rate (which in general can be a symmetric matrix), and $t = 0, 1, 2, \ldots$ are the time steps. In gradient flow, we can define a smooth curve in the parameter space from a choice of initial values to the limiting local minimum without discretizing over time. The curve is a function of a continuous time variable $t \in \mathbb{R}$, and velocity of this curve at any point is equal to the gradient of the loss function, scaled by the negative of the learning rate. In other words, the dynamics of the parameters under GF are given by:

$$\dot{\boldsymbol{\theta}}(t) = d\boldsymbol{\theta}(t)/dt = -\varepsilon \nabla_{\boldsymbol{\theta}(t)} L. \tag{2.10}$$

From an initialization $\boldsymbol{\theta}(0)$ at $t = 0$, GF defines a trajectory $\boldsymbol{\theta}(t) \in \mathsf{Param}$ for $t \in \mathbb{R}_{>0}$, which limits to a critical point. In this way, GF is a continuous version of GD.

Conserved quantities.

A *conserved quantity* of GF is a function $Q : \mathsf{Param} \to \mathbb{R}$ such that the value of Q at any two time points $s, t \in \mathbb{R}_{>0}$ along a GF trajectory is the same: $Q(\boldsymbol{\theta}(s)) = Q(\boldsymbol{\theta}(t))$. In other words, we have $dQ(\boldsymbol{\theta}(t))/dt = 0$. Note that, if $f : \mathbb{R} \to \mathbb{R}$ is any function, and Q is a conserved quantity, then the composition $f \circ Q$ is also a conserved quantity. Several conserved quantities of GF have appeared in the literature, most notably layer imbalance $Q_{\text{imb}} \equiv \|W_i\|^2 - \|W_{i-1}\|^2$ [38] for each pair of successive feedforward linear layers ($\sigma(x) = x$), and its full matrix version $Q_i = W_i^T W_i - W_{i-1} W_{i-1}^T$.

We now propose a generalization of the layer imbalance by associating a conserved

quantity to any infinitesimal symmetry. As in Section 2.3.2, suppose a matrix Lie group G acts linearly on the parameter space. Then, from equation 2.6, we have the identity $\langle \nabla_\theta L, M \cdot \boldsymbol{\theta} \rangle = 0$ for any element M in the Lie algebra $\mathfrak{g}$. Using the gradient flow dynamics equation 2.10, this identity becomes:

$$\left\langle \varepsilon^{-1} \dot{\boldsymbol{\theta}} \ , \ M \cdot \boldsymbol{\theta} \right\rangle = 0 \tag{2.11}$$

In other words, the velocity at any point of a gradient flow curve is orthogonal to the infinitesimal action. For simplicity, we set the learning rate to the identity: $\varepsilon = I$ (all results generalize to symmetric ε.) The following proposition (whose proof is elementary and well-known) provides a way of 'integrating' equation equation 2.11, in the appropriate sense, in order to obtain conserved quantities:

Proposition 2.5.1. *Suppose the action of G on* Param *is linear*[7] *and leaves L invariant. For any $M \in \mathfrak{g}$, there is a conserved quantity $Q_M : \mathsf{Param} \to \mathbb{R}$ given by $Q_M(\boldsymbol{\theta}) = \langle \boldsymbol{\theta} \ , \ M \cdot \boldsymbol{\theta} \rangle$.*

While Proposition 2.5.1 directly links the infinitesimal action to conserved quantities, it has the limitation that the conserved quantity corresponding to an anti-symmetric matrix $M = -M^T$ in $\mathfrak{g}$ is constantly zero, and we do not obtain meaningful conserved quantities. Instead, we can only conclude that flow curves satisfy the differential equation equation 2.11. Fixing a basis $(\theta^1, \cdots \theta^d)$ for $\mathsf{Param} \simeq \mathbb{R}^d$, this equation becomes $\sum_{i<j} M_{ij} r_{ij}^2 \dot{\phi}_{ij} \equiv 0$ where (r_{ij}, ϕ_{ij}) are the polar coordinates for the point $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \in \mathbb{R}^2$ (see Appendix A.2.9). In summary, we find:

$M \in \mathfrak{g}$	symmetric M	anti-symmetric M
differential equation	conserved quantity	differential equation
$\dot{\theta}^T M \theta = 0$	$Q_M(\theta) = \theta^T M \theta$	$\sum_{i<j} m_{ij} r_{ij}^2 \dot{\phi}_{ij} \equiv 0$

[7]For simplicity, we also assume that G is closed under taking transposes, and acts faithfully on the parameter space. These assumptions generally hold in practice; see Appendix A.2 for a version with fewer assumptions.

Conserved quantities parametrize symmetry flat directions.

We observe that applying a symmetry changes the values of the conserved quantities Q_M (Figure 2.1). Indeed, for $M \in \mathfrak{g}$ and $g \in G$, we have $Q_M(g \cdot \boldsymbol{\theta}) = Q_{g^T M g}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \mathsf{Param}$, so applying the group action[8] transforms the conserved quantity Q_M to $Q_{g^T M g}$. As discussed in Section 2.3, applying g to a minimum $\boldsymbol{\theta}^*$ of L yields another minimum $g \cdot \boldsymbol{\theta}^*$; hence applying symmetries leads to a partial parameterization of flat minima. Note that, in general, we may lack sufficient number of Q_M to fully parameterize a flat minimum. For example, in the linear network UVx, $G = \mathrm{GL}_h$ and flat minima generically have h^2 dimensions, whereas the number of independent nonzero Q_M is $h(h+1)/2$, which is the dimension of the space of symmetric matrices $M = M^T$ in $\mathfrak{gl}_h$.

In gradient descent, the values of these conserved quantities may change due to the time discretization. However, the change in Q is expected to be small. For example, in two-layer linear networks, the change of Q is bounded by the square of learning rate. Appendix A.4 contains derivations and empirical observations of the magnitude of change in Q.

Relation to Noether's theorem.

In physics, Noether's theorem [112] states that continuous symmetries give rise to conserved quantities. Recently, [134] showed that Noether's theorem can also be applied to GD by approximating it as a *second order* GF. We show that in the limit where the second order GF reduces to first order GF equation 2.10, results from Noether's theorem reduce to our conservation law $\left\langle \overline{M}_{\boldsymbol{\theta}}, \nabla L \right\rangle = 0$ equation 2.6. In short, using Noether's theorem, the conserved Noether current is $J_M = e^{t/\tau} J_{0M}$ with $J_{0M} = \left\langle \overline{M}_{\boldsymbol{\theta}}, \varepsilon^{-1} \dot{\boldsymbol{\theta}} \right\rangle$. In the limit $\tau \to 0$, using equation 2.10, $J_{0M} = \left\langle \overline{M}_{\boldsymbol{\theta}}, \nabla L \right\rangle = 0$ and the conservation $dJ_M/dt = 0$ implies $J_{0M} = 0$, meaning we recover equation 2.6. Details appear in Appendix A.1.

[8] Note that this procedure only works if $g^T M g$ belongs to $\mathfrak{g}$, which is the case the examples we consider.

Examples.

We present examples of conserved quantities for two-layer neural networks. all of which directly generalize to the multi-layer case. See Appendix A.2.9 for full derivations (which heavily rely on properties of the trace). We adopt the notation of Section 2.3.3.

Example 2.5.2 (General equivariant activation)**.** *Suppose σ is equivariant under a linear action of a subgroup $G \subseteq \mathrm{GL}_h(\mathbb{R})$, so that $\pi(g)\sigma(z) = \sigma(gz)$. Then the two-layer network $F(z) = U\sigma(Vz)$ is invariant under G, as is the loss function. For symmetric $M \in \mathfrak{g}$, Proposition 2.5.1 yields the following conserved quantity:*

$$Q_M : \mathsf{Param} \to \mathbb{R}, \qquad Q_M(U,V) = \mathrm{Tr}\left[V^T M V\right] - \mathrm{Tr}\left[U^T U d\pi(M)\right] \tag{2.12}$$

Indeed, this follows from the fact that $M \cdot (U,V) = (-Ud\pi(M), MV)$, as in equation 2.5.

Example 2.5.3 (Imbalance in linear layers)**.** *Suppose the network is linear. Then $\sigma(z) = z$ and the loss is invariant under $\mathrm{GL}_h(\mathbb{R})$. For symmetric M we have the conserved quantity $Q_M(U,V) = \mathrm{Tr}\left[(VV^T - U^T U)M\right]$. Moreover, each component of the matrix $VV^T - U^T U$ is conserved.*

Example 2.5.4 (Homogeneous activation under scaling)**.** *Suppose σ is a homogeneous activation of degree α. Let $G = (\mathbb{R}_{>0})^h$ be the positive rescaling group, so that $\sigma(gz) = g^\alpha \sigma(z)$ for any $g \in G$ and $z \in \mathbb{R}^h$. Note that the Lie algebra of G consists of all diagonal matrices in $\mathfrak{gl}_h$, so that, in particular, each $M \in \mathfrak{g}$ is symmetric. Since $d\pi(M) = \alpha M$ for any $M \in \mathfrak{g}$, we obtain the conserved quantity $Q_M(U,V) = \mathrm{Tr}\left[(VV^T - \alpha U^T U)M\right]$. Using the basis $M = E_{kk}$, we see that $\mathbf{Q} = \mathrm{diag}\left[VV^T - \alpha U^T U\right]$ is conserved (here, $\mathrm{diag}[A]$ is the leading diagonal). Special cases of this are LeakyReLU and ReLU with $\alpha = 1$.*

Example 2.5.5 (Radial rescaling activations)**.** *Let σ be such a radial rescaling activation. As in Section 2.3.3, the orthogonal group $G = O(h)$ is a symmetry of L. The Lie algebra $\mathfrak{g} = \mathfrak{so}_h$ comprises anti-symmetric matrices, and so Proposition 2.5.1 yields no non-trivial conserved*

quantities. However, using the canonical basis of $\mathfrak{g} = \mathfrak{so}_h$ *given by* $E_{[kl]} = E_{kl} - E_{lk}$ *(so* $[kl]$ *indicates anti-symmetrized indices), one uses equation equation 2.11 to deduce the following novel result (see Appendix A.2.9):*

Theorem 2.5.6. *When* σ *is a radial rescaling activation, we have:*

$$V\dot{V}^T - \dot{V}V^T + U^T\dot{U} - \dot{U}^T U = 0 \tag{2.13}$$

for any $(U,V) \in \mathsf{Param}$*, where the dots indicate derivatives with respect to gradient flow.*

Expanding the (k,l) *entry of the matrix on the left-hand-side of equation 2.13, we obtain:* $\sum_{s=1}^{n} r_{U,s;kl}^2 \dot{\phi}_{U,s;kl} + \sum_{s=1}^{m} r_{V^T,s;kl}^2 \dot{\phi}_{V^T,s;kl} = 0$*, where* $(r_{U,s;kl}, \phi_{U,s;kl})$ *and* $(r_{V^T,s;kl}, \phi_{V^T,s;kl})$ *are the 2D polar coordinates of the points* (U_{sk}, U_{sl}) *and* (V_{sk}^T, V_{sl}^T)*. This is analogous to the "angular momentum" in 2D, that is:* $x \wedge \dot{x} = r^2\dot{\phi}$*. Intuitively, Theorem 2.5.6 implies that in every 2D plane* (k,l)*, the angular momenta of the rows of* U *and the columns of* V *sum to zero. These results also apply to linear networks* $F(x) = UVx$*, since rotational symmetries are linear.*

2.6 Distribution of Conserved Quantities under Xavier Initialization

We first consider a linear two-layer neural network UVX, where $U \in \mathbb{R}^{m\times h}, V \in \mathbb{R}^{h\times n}$, and $X \in \mathbb{R}^{n\times k}$. We choose the following form of the conserved quantity:

$$Q = \frac{1}{2}\,\mathrm{Tr}[U^T U - VV^T]. \tag{2.14}$$

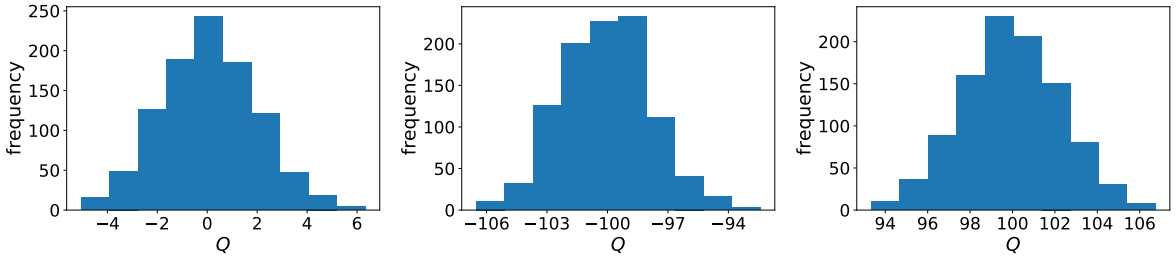
Xavier initialization keeps the variance of each layer's output the same as the variance of the input. Under Xavier initialization [53], each element in a given layer is initialized independently, with mean 0 and variance equal to the inverse of the layer's input dimension:

$$U_{ij} = \mathcal{N}\left(0, \frac{1}{h}\right) \qquad\qquad V_{ij} = \mathcal{N}\left(0, \frac{1}{n}\right) \tag{2.15}$$

The expected value of Q is

$$\mathbb{E}[[]\,Q] = Var(U_{ij}) \times m \times h + Var(V_{ij}) \times h \times n = m - h. \tag{2.16}$$

Figure 2.2 shows the distribution of Q for 2-layer linear NN with different layer dimensions. For each dimension tuples (m,h,n), we constructed 1000 sets of parameters using Xavier initialization. The centers of the distributions of Q match Eq. equation 2.16.



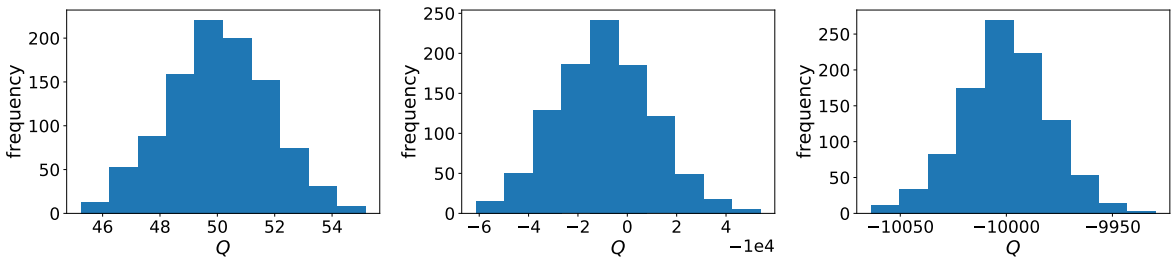
(a) $m,h,n = 100,100,100.$ (b) $m,h,n = 100,200,100.$ (c) $m,h,n = 200,100,100.$

Figure 2.2. Distribution of Q for 2-layer linear NN with different layer dimensions.

Next, we consider the nonlinear two-layer neural network $U\sigma(VX)$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is an element-wise activation function. For simplicity, we assume whitened input ($X = I$). We choose the following form of the conserved quantity:

$$Q = \frac{1}{2}\mathrm{Tr}[U^T U] - \sum_{a,j} \int_0^{V_{aj}} dx \frac{\sigma(x)}{\sigma'(x)} \tag{2.17}$$

Figure 2.3 shows the distribution of Q for 2-layer linear NN with different nonlinearities, each with 1000 sets of parameters created under Xavier initialization. The shapes of the distributions are similar to that of linear networks. The value of Q is usually concentrated around a small range of values. Since the range of Q is unbounded, the Xavier initialization limits the model to a small part of the global minimum.



(a) ReLU. (b) Tanh. (c) Sigmoid.

Figure 2.3. Distribution of Q for 2-layer linear NN with different nonlinearities, with parameter dimensions $m = h = n = 100$.

2.7 Conserved Quantity and Convergence Rate

The values of conserved quantities are unchanged throughout the gradient flow. Since the conserved quantities parameterize trajectories, initializing parameters with certain conserved quantity values accelerates convergence. For two-layer linear reparametrization, [135] derived the explicit relation between layer imbalance and convergence rate. We derive the relation between conserved quantities and convergence rate for two example optimization problems and provide numerical evidence that initializing parameters with optimal conserved quantity values accelerates convergence.

2.7.1 Example 1: Ellipse

We first show that the convergence rate is related to the conserved quantity in a toy optimization problem. Consider the following loss function with $a \in \mathbb{R}$:

$$\begin{aligned} L(w_1, w_2) &= w_1^2 + a w_2^2 \\ \nabla L &= (2w_1, 2aw_2) \end{aligned} \tag{2.18}$$

Assuming gradient flow,

$$\frac{dw_1}{dt} = -\nabla_{w_1} L = -2w_1 \qquad \frac{dw_2}{dt} = -\nabla_{w_2} L = -2aw_2 \tag{2.19}$$

Then w_1, w_2 are governed by the following differential equations:

$$w_1(t) = w_{1_0} e^{-2t} \qquad\qquad w_2(t) = w_{2_0} e^{-2at} \tag{2.20}$$

where w_{1_0}, w_{2_0} are initial values of w_1 and w_2. We can find conserved quantities by using an ansatz $Q = f(w_1^i w_2^k)$ and solving $\nabla Q \cdot \nabla L = 0$ for i, k. Below we use the following form of conserved quantity:

$$Q = \frac{w_1^{2a}}{w_2^2} = \frac{w_{1_0}^{2a}}{w_{2_0}^2} \tag{2.21}$$

To show the effect of Q on the convergence rate, we fix $L(0)$ and derive how Q affects $L(t)$. Let $L(0) = w_{1_0}^2 + a w_{2_0}^2 = L_0$. Let w_{2_0} continue to be an independent variable. Then $w_{1_0}^2 = L_0 - a w_{2_0}^2$. Substitute in $w_{1_0}^2$, the loss at time t is

$$L(t) = w_1(t)^2 + a w_2(t)^2 = (L_0 - a w_{2_0}^2) e^{-4t} + a w_{2_0}^2 e^{-4at} \tag{2.22}$$

and Q becomes

$$Q = \frac{w_{1_0}^{2a}}{w_{2_0}^2} = \frac{(L_0 - a w_{2_0}^2)^a}{w_{2_0}^2} \tag{2.23}$$

The derivative of L in the direction of Q is

$$\begin{aligned}
\partial_Q L(t) &= \frac{dL(t)}{dw_{2_0}} \frac{dw_{2_0}}{dQ} = \frac{dL(t)}{dw_{2_0}} \left(\frac{dQ}{dw_{2_0}} \right)^{-1} \\
&= \left(-2a w_{2_0} e^{-4t} + 2a w_{2_0} e^{-4at} \right) \left(\frac{a(L_0 - a w_{2_0}^2)^{a-1}(-2a w_{2_0}) w_{2_0}^2 - 2 w_{2_0} (L_0 - a w_{2_0}^2)^a}{w_{2_0}^4} \right)^{-1} \\
&= \frac{\left(-2a w_{2_0} e^{-4t} + 2a w_{2_0} e^{-4at} \right) w_{2_0}^4}{a(L_0 - a w_{2_0}^2)^{a-1}(-2a w_{2_0}) w_{2_0}^2 - 2 w_{2_0} (L_0 - a w_{2_0}^2)^a} \\
&= \frac{2a w_{2_0}^5 \left(e^{-4at} - e^{-4t} \right)}{2 w_{2_0} (L_0 - a w_{2_0}^2)^{a-1} \left(-a^2 w_{2_0}^2 - (L_0 - a w_{2_0}^2) \right)}
\end{aligned} \tag{2.24}$$

In general, $\partial_Q L(t) \neq 0$, meaning that the loss at time t depends on Q. Since we have fixed the initial loss, the convergence rate $L(t) - L(0)$ also depends on Q. Special cases where $\partial_Q L(t) = 0$ include $a = 1$ (circle), $a = 0$ (collapsed dimension), and certain initializations such as $w_{2_0} = 0$ (local maximum of gradient magnitude).

2.7.2 Example 2: Radial Activation Functions

In this example, we find the conserved quantities and their relation with convergence rate for two-layer reparametrization with radial activation functions under spectral initialization.

Define radial function $g : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ as

$$g(W)_{ij} = h(|W_i|)\, W_{ij}, \tag{2.25}$$

where $|W_i| = \left(\sum_k W_{ik}^2\right)^{\frac{1}{2}}$ is the norm of the i^{th} row of W, and $h : \mathbb{R} \to \mathbb{R}$ outputs a scalar.

Consider the following objective:

$$\text{argmin}_{U,V}\{L(U,V) = \frac{1}{2}\|Y - Ug(V^T)\|_F^2\} \tag{2.26}$$

with spectral initializations

$$U_0 = \Phi\overline{U}_0, V_0 = \Psi\overline{V}_0,$$

where Φ, Ψ come from the singular value decomposition $Y = \Phi\Sigma_Y\Psi^T$, and $\overline{U}_0, \overline{V}_0$ are random diagonal matrices.

Proposition 2.7.1. *Under the gradient flow $U = -\nabla_U L$ and $V = -\nabla_V L$, the following quantity is an invariant:*

$$Q = \frac{1}{2}\text{Tr}[U^T U] - \sum_i \int_{x_0}^{\bar{V}_{ii}} dx \frac{g(x)}{g'(x)} \tag{2.27}$$

Proof. Since g is a radial function on rows and Ψ^T is an orthogonal matrix, $g(\overline{V}^T\Psi^T) = g(\overline{V}^T)\Psi^T$. With spectral initialization, the loss function can be reduced to only involving diagonal matrices:

$$\begin{aligned} L &= \frac{1}{2}\|Y - Ug(V^T)\|_F^2 \\ &= \frac{1}{2}\|\Phi\Sigma\Psi^T - \Phi\overline{U}g[(\Psi\overline{V})^T]\|_F^2 \\ &= \frac{1}{2}\|\Phi\Sigma\Psi^T - \Phi\overline{U}g(\overline{V}^T)\Psi^T\|_F^2 \\ &= \frac{1}{2}\|\Phi\left(\Sigma - \overline{U}g(\overline{V}^T)\right)\Psi^T\|_F^2 \\ &= \frac{1}{2}\|\Sigma - \overline{U}g(\overline{V}^T)\|_F^2 \end{aligned} \tag{2.28}$$

Since $\overline{V}$ is a diagonal matrix, g is now an element wise function on $\overline{V}$. Let $\overline{W} = \overline{U}g(\overline{V}^T)$. The gradients for $\overline{U}$ and $\overline{V}$ are

$$\begin{aligned} \frac{\partial L}{\partial \overline{U}} &= \nabla_{\overline{W}} L g(\overline{V})^T \\ \frac{\partial L}{\partial \overline{V}} &= \nabla_{\overline{W}} L^T \overline{U} \odot g'(\overline{V}) \end{aligned} \tag{2.29}$$

where $g'(x) = dg(x)/dx$ is the derivative of the nonlinearity. Additionally, since L does not depend on Φ and Ψ,

$$\frac{\partial L}{\partial \Phi} = \frac{\partial L}{\partial \Psi} = 0 \tag{2.30}$$

Since the rows of Φ, Ψ are orthogonal,

$$\begin{aligned} \frac{\partial L}{\partial U} &= \frac{\partial L}{\partial \overline{U}}\Phi^T = \nabla_{\overline{W}} L g(\overline{V})^T \Phi^T \\ \frac{\partial L}{\partial V} &= \frac{\partial L}{\partial \overline{V}}\Psi^T = \left(\nabla_{\overline{W}} L^T \overline{U} \odot g'(\overline{V})\right)\Psi^T \end{aligned} \tag{2.31}$$

Φ and Ψ are not changed in gradient flow, so $\frac{\partial Q}{\partial U} = \frac{\partial Q}{\partial \overline{U}}\Phi^T$ and $\frac{\partial Q}{\partial V} = \frac{\partial Q}{\partial \overline{V}}\Psi^T$. Define

inner product on matrices as $\langle X, Y \rangle = \text{Tr}[X^T Y]$. For Q to be a conserved quantity, we need $\langle \nabla L, \nabla Q \rangle = 0$:

$$\begin{aligned}
\langle \nabla L, \nabla Q \rangle &= \langle \frac{\partial L}{\partial U}, \frac{\partial Q}{\partial U} \rangle + \langle \frac{\partial L}{\partial V}, \frac{\partial Q}{\partial V} \rangle \\
&= \langle \nabla_{\overline{W}} L g(\overline{V})^T \Phi^T, \frac{\partial Q}{\partial \overline{U}} \Phi^T \rangle + \langle \left(\nabla_{\overline{W}} L^T \overline{U} \odot g'(\overline{V}) \right) \Psi^T, \frac{\partial Q}{\partial \overline{V}} \Psi^T \rangle \\
&= \langle \nabla_{\overline{W}} L g(\overline{V})^T, \frac{\partial Q}{\partial \overline{U}} \rangle + \langle \left(\nabla_{\overline{W}} L^T \overline{U} \odot g'(\overline{V}) \right), \frac{\partial Q}{\partial \overline{V}} \rangle \\
&= \text{Tr} \left[\partial_{\overline{U}^T} Q \nabla_{\overline{W}} L g(\overline{V})^T + \overline{U}^T \nabla_{\overline{W}} L (\partial_V Q \odot g'(\overline{V})) \right] = 0
\end{aligned} \tag{2.32}$$

Following the same procedure as for elementwise functions, to have a Q which satisfies equation 2.32 it is sufficient to have

$$\frac{\partial Q}{\partial \overline{U}_{ia}} = f(\overline{U}, \overline{V}) \overline{U}_{ia} \qquad \frac{\partial Q}{\partial \overline{V}_{aj}} g'(\overline{V})_{aj} = -f(\overline{U}, \overline{V}) g(\overline{V})_{aj} \qquad f(\overline{U}, \overline{V}) \in \mathbb{R} \tag{2.33}$$

For simplicity, let $f(\overline{U}, \overline{V}) = 1$. Then, equation 2.33 is satisfied by

$$Q = \frac{1}{2} \text{Tr}[\bar{U}^T \bar{U}] - \sum_i \int_{x_0}^{\bar{V}_{ii}} dx \frac{g(x)}{g'(x)} \tag{2.34}$$

□

[135] shows that the conserved quantity Q appears as a term in the convergence rate of the matrix factorization gradient flow. We observe a similar relationship between Q and convergence rate when the loss function is augmented with a radial activation function, as shown in the following proposition.

Proposition 2.7.2. *Consider the objective function and spectral initialization defined in Proposition 2.7.1. Let $h(|W_i|) = |W_i|^{-2}$, and $X = U g(V^T) = \Phi \Sigma_X \Psi^T$. Then, the eigencomponent of X*

approaches the corresponding eigencomponent of Y at a rate of

$$\dot{\sigma}_i^X = \frac{1}{\lambda_i}(\sigma_i^Y - \sigma_i^X)(\sigma_i^{X2} + 1)^2, \tag{2.35}$$

where $\sigma_i^X = diag(\Sigma_X)_i$, $\sigma_i^Y = diag(\Sigma_Y)_i$, and $\lambda_i = \bar{U}_{ii}^2 + \bar{V}_{ii}^2$ are conserved quantities.

Proof. Similar to [135], components can be decoupled, and we have a set of differential equations on scalars:

$$\begin{aligned} \dot{\bar{u}}_i &= [\sigma_i^Y - \bar{u}_i g(\bar{v}_i)] g(\bar{v}_i) \\ \dot{\bar{v}}_i &= [\sigma_i^Y - \bar{u}_i g(\bar{v}_i)] \bar{u}_i \frac{dg(\bar{v}_i)}{d\bar{v}_i} \end{aligned} \tag{2.36}$$

We also have

$$\dot{g}(\bar{v}_i) = \frac{dg}{d\bar{v}_i}\frac{d\bar{v}_i}{dt} = [\sigma_i^Y - \bar{u}_i g(\bar{v}_i)] \bar{u}_i \left(\frac{dg(\bar{v}_i)}{d\bar{v}_i}\right)^2. \tag{2.37}$$

Let $\sigma_i^X = \bar{u}_i g(\bar{v}_i)$. Then

$$\begin{aligned} \dot{\sigma}_i^X &= \dot{\bar{u}}_i g(\bar{v}_i) + \bar{u}_i \dot{g}(\bar{v}_i) \\ &= \left[\sigma_i^Y - \bar{u}_i g(\bar{v}_i)\right] \left[g(\bar{v}_i)^2 + \bar{u}_i^2 \left(\frac{dg(\bar{v}_i)}{d\bar{v}_i}\right)^2 \right]. \end{aligned} \tag{2.38}$$

Since $\bar{V}$ is a diagonal matrix, g is now an element wise function on $\bar{V}$. Specifically, $g(\bar{v}_i) = \frac{1}{\bar{v}_i}$. According to Proposition 2.7.1, the following quantity is invariant:

$$\frac{1}{2}\bar{u}_i^2 - \int dx \frac{g(x)}{g'(x)} = \frac{1}{2}\bar{u}_i^2 - \int dx \frac{\bar{v}_i^{-1}}{-\bar{v}_i^{-2}} = \frac{1}{2}\bar{u}_i^2 + \frac{1}{2}\bar{v}_i^2 \tag{2.39}$$

Since any function of the invariant is also invariant, we will use the following form:

$$Q = \bar{U}^T \bar{U} + \bar{V}^T \bar{V}, \tag{2.40}$$

and define

$$\lambda_i = Q_{ii} = \overline{u}_i^2 + \overline{v}_i^2 \tag{2.41}$$

Using the g that we defined,

$$\sigma_i^X = \overline{u}_i g(\overline{v}_i) = \overline{u}_i \overline{v}_i^{-1}. \tag{2.42}$$

In order to relate σ^X and Q, we first write $\overline{u}_i$ and $\overline{v}_i$ as functions of σ_i^X ad Q using equation 2.41 and equation 2.42:

$$\overline{u}_i^2 = \frac{\lambda_i \sigma_i^{X\,2}}{\sigma_i^{X\,2}+1}, \qquad\qquad \overline{v}_i^2 = \frac{\lambda_i}{\sigma_i^{X\,2}+1}. \tag{2.43}$$

Then, substitute $\overline{u}_i$, $\overline{v}_i$, $g(\overline{v}_i)$, and $\frac{dg(\overline{v}_i)}{d\overline{v}_i}$ into equation 2.38, and we have

$$\begin{aligned}
\dot{\sigma}_i^X &= [\sigma_i - \overline{u}_i g(\overline{v}_i)] \left[g(\overline{v}_i)^2 + \overline{u}_i^2 \left(\frac{dg(\overline{v}_i)}{d\overline{v}_i} \right)^2 \right] \\
&= \left[\sigma_i^Y - \overline{u}_i g(\overline{v}_i) \right] \left[\left(\frac{1}{\overline{v}_i} \right)^2 + \overline{u}_i^2 \left(-\overline{v}_i^{-2} \right)^2 \right] \\
&= \left[\sigma_i^Y - \overline{u}_i g(\overline{v}_i) \right] \left[(\overline{v}_i^2)^{-1} + \overline{u}_i^2 (\overline{v}_i^2)^{-2} \right] \\
&= \left[\sigma_i^Y - \sigma_i^X \right] \left[\left(\frac{\lambda_i}{\sigma_i^{X\,2}+1} \right)^{-1} + \frac{\lambda_i \sigma_i^{X\,2}}{\sigma_i^{X\,2}+1} \left(\frac{\lambda_i}{\sigma_i^{X\,2}+1} \right)^{-2} \right] \\
&= \left[\sigma_i^Y - \sigma_i^X \right] \left[\frac{\sigma_i^{X\,2}+1}{\lambda_i} + \frac{\sigma_i^{X\,2}(\sigma_i^{X\,2}+1)}{\lambda_i} \right] \\
&= \left[\sigma_i^Y - \sigma_i^X \right] \left[\frac{\sigma_i^{X\,4} + 2\sigma_i^{X\,2} + 1}{\lambda_i} \right] \\
&= \frac{1}{\lambda_i} (\sigma_i^Y - \sigma_i^X)(\sigma_i^{X\,2}+1)^2
\end{aligned} \tag{2.44}$$

□

Proposition 2.7.2 relates the rate of change in parameters $\dot{\sigma}_i^X$ and the conserved quantity λ_i. To get a more explicit expression of how λ_i affects convergence rate, we will derive a bound for $|\sigma_i^Y - \sigma_i^X|$, which describes the distance between trainable parameters to their desired value.

Proposition 2.7.3. *The difference between the singular values of $Ug(V^T)$ and Y is bounded by*

$$|\sigma_i^X - \sigma_i^Y| \leq |\sigma_i^X(0) - \sigma_i^Y| e^{-\frac{t}{\lambda_i}}. \tag{2.45}$$

Proof. Note that

$$\dot{\sigma_i^X} = \frac{1}{\lambda}(\sigma_i^Y - \sigma_i^X)({\sigma_i^X}^2 + 1)^2 \geq \frac{1}{\lambda_i}(\sigma_i^Y - \sigma_i^X) \tag{2.46}$$

Consider the following two differential equations, with same initialization $a(0) = b(0)$:

$$\begin{aligned} \dot{a} &= \frac{1}{\lambda}(\sigma - a)(a^2+1)^2 \\ \dot{b} &= \frac{1}{\lambda}(\sigma - b) \end{aligned} \tag{2.47}$$

In these equations, both a and b moves from $a(0) = b(0)$ to σ monotonically. Since $\dot{a} \geq \dot{b}$ at every $a = b$, a will always be closer to σ than b does. We can explicitly solve for b, which yields $b(t) = \sigma + (b(0) - \sigma)e^{-\frac{t}{\lambda}}$. Then the distance between b and σ is $|b - \sigma| = |b(0) - \sigma| e^{-\frac{t}{\lambda}}$. Using $|b - \sigma|$, we can bound $|a - \sigma|$:

$$|a - \sigma| \leq |b - \sigma| = |b(0) - \sigma| e^{-\frac{t}{\lambda}} \tag{2.48}$$

Therefore,

$$|\sigma_i^X - \sigma_i^Y| \leq |\sigma_i^X(0) - \sigma_i^Y| e^{-\frac{t}{\lambda_i}} \tag{2.49}$$

□

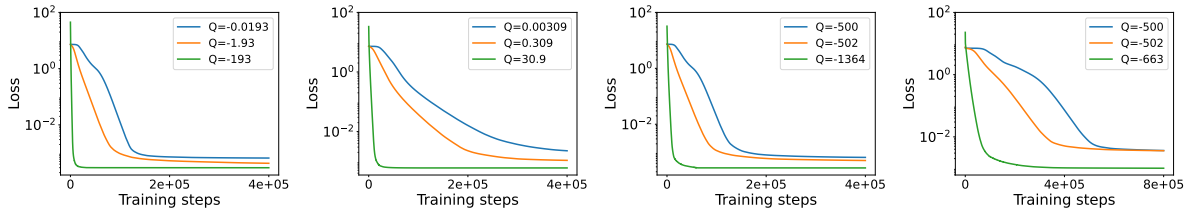
Since λ is a conserved quantity, its value set at initialization remains unchanged throughout the gradient flow. Therefore, we are able to optimize the convergence rate by choosing a favorable value for λ at initialization. In this example, smaller λ_i's lead to faster convergence.

2.7.3 Experiments

We compare the convergence rate of two-layer networks initialized with different Q values. We run gradient descent on two-layer networks with whitened input with the following objective

$$\text{argmin}_{U,V}\{L(U,V) = \|Y - U\sigma(V^T)\|_F^2\} \tag{2.50}$$

where σ is the identity function, ReLU, sigmoid, or tanh. Matrices $Y \in \mathbb{R}^{5\times 10}$, $U \in \mathbb{R}^{5\times 50}$ and $V \in \mathbb{R}^{10\times 50}$ have random Gaussian initialization with zero mean. We repeat the gradient descent with learning rate 0.1, 0.01, and 0.001. The learning rate is set to 10^{-3}, as we do not observe significant changes in the shape of learning curves at smaller learning rates. U and V are initialized with different variance, which leads to different initial values of Q.



(a) Linear. (b) ReLU. (c) Tanh. (d) Sigmoid.

Figure 2.4. Training curves of two-layer networks initialized with different Q. The value of Q affects convergence rate.

As shown in Fig.2.4, the number of steps required for the loss curves to drop to near convergence level is correlated with Q in both linear and element-wise nonlinear networks. This result provides empirical evidence that initializing parameters with optimal values for Q accelerates convergence.

We then demonstrate the effect of conserved quantity values on the convergence rate of radial neural networks. Fig.2.5 shows the training curve for loss function defined in Proposition 2.7.2. We initialize parameters $U \in \mathbb{R}^{5\times 5}$ and $V \in \mathbb{R}^{10\times 5}$ with 4 different values of Q and the learning rate is set to 10^{-5}. As predicted in Eq. 2.45, convergence is faster when $Q = \text{Tr}[U^T U + V^T V]$ is small.

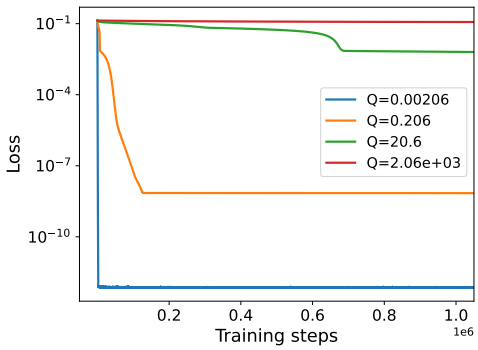


Figure 2.5. Training curve for the loss function defined in Proposition 2.7.2. Smaller value of $Q = \text{Tr}[U^T U + V^T V]$ at initialization leads to faster convergence.

2.8 Conserved Quantity and Generalization Ability

Conserved quantities parameterize the minimum of neural networks and are related to the eigenvalues of the Hessian at minimum. Recent theory and empirical studies suggest that sharp minimum do not generalize well [64, 76, 113]. Explicitly searching for flat minimum has been shown to improve generalization bounds and model performance [28, 46, 77]. We derive their relationship for the simplest two-layer network, and show empircally that conserved quantity values affect sharpness. Like convergence rate, a systematic study of the relationship between conserved quantity and generalization ability of the solution is an interesting future direction.

2.8.1 Example: Two-Layer Linear Network with 1D Parameters

We again consider the two-layer linear network with loss $L = \frac{1}{2}\|Y - UVX\|^2$. For simplicity, we work with one dimensional parameters $U, V \in \mathbb{R}$ and assume $X = Y = 1$ in this

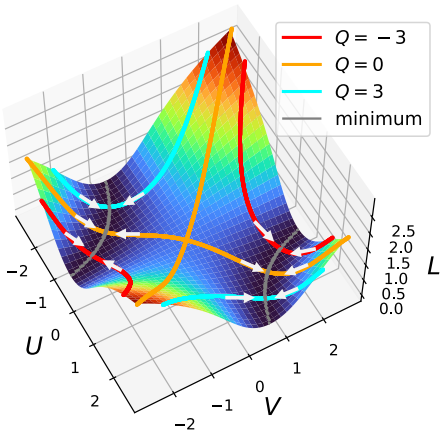


Figure 2.6. Gradient flow for $L(U,V) = \frac{1}{2}\|Y - UVX\|^2$, where $U,V \in \mathbb{R}$, $Y = 2$, and $X = 1$. Trajectories corresponding to different values of Q intersect the minima at different points.

example. We show that at the point to which the gradient flow converges, the eigenvalues of the Hessian are related to the value of the conserved quantity.

The gradients and Hessian of L are

$$\nabla L = \begin{bmatrix} -(Y - UVX)VX \\ -(Y - UVX)UX \end{bmatrix} \qquad \mathscr{H} = \begin{bmatrix} V^2X^2 & -YX + 2UVX^2 \\ -YX + 2UVX^2 & U^2X^2 \end{bmatrix} \tag{2.51}$$

At the minima, U,V are related by $UVX = Y$. Recall that $Q = U^2 - V^2$ is a conserved quantity. From the above two equations, we can write U,V as functions of Q. Taking the solution $U = \sqrt{\frac{1}{2}(Q + \sqrt{Q^2+4})}, V = \sqrt{\frac{1}{2}(-Q + \sqrt{Q^2+4})}$ and substitute in $X = Y = 1$, we have

$$\mathscr{H} = \begin{bmatrix} \frac{1}{2}(-Q + \sqrt{Q^2+4}) & 1 \\ 1 & \frac{1}{2}(Q + \sqrt{Q^2+4}) \end{bmatrix}, \tag{2.52}$$

and the eigenvalues of $\mathscr{H}$ are

$$\lambda_1 = 0, \qquad \lambda_2 = 2\sqrt{Q^2+4}. \tag{2.53}$$

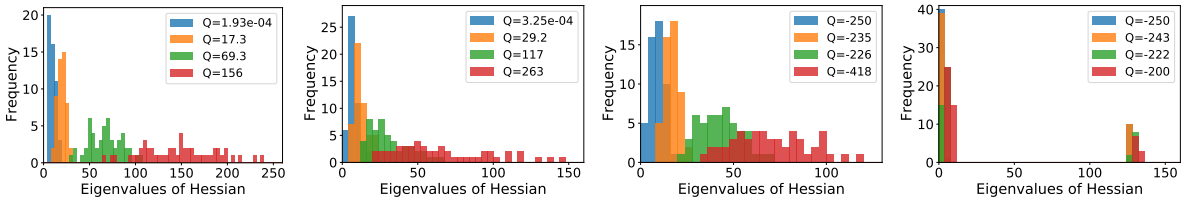
We have shown that Q is related the eigenvalues of the Hessian at the minimum. Since

the eigenvalues determines the curvature, Q also determines the sharpness of the minimum, which is believed to related to model's generalization ability. The result in this example can also be observed in Figure 2.1, where the minimum of the $Q = 0$ trajectory lies at the least sharp point of the loss valley.

2.8.2 Experiments: Two-Layer Networks

The goal of this section is to explore the relation between Q and the sharpness of the trained model. We measure sharpness by the magnitude of the eigenvalues of the Hessian, which are related to the curvature at the minima. We use the same loss function equation 2.50 in Section 2.7.3. The parameters are $U \in \mathbb{R}^{10\times 50}$ and $V \in \mathbb{R}^{5\times 50}$, each initialized with zero mean and various standard deviations that lead to different Q's. We first train the models using gradient descent. We then use the vectorized parameters in the trained model to compute the eigenvalues of the Hessian.

The linear model extends the example in Section 2.8.1 to higher dimension parameter spaces. 700 out of the 750 eigenvalues are around 0 (with magnitude $\leq 10^{-3}$), which verifies the dimension of the minima in Proposition A.2.9. After removing the small eigenvalues, the center of the eigenvalue distribution correlates positively with the value of Q (Figure 2.7(a)). In models with nonlinear activations, Q is still related to eigenvalue distributions, although the relations seem to be more complicated.



(a) Linear. (b) ReLU. (c) Tanh. (d) Sigmoid.

Figure 2.7. Eigenvalues of the Hessian from trained models initialized with different conserved quantity values (Q). The distribution of the eigenvalues and the value of Q appear to be related.

2.9 Ensemble Models

In neural networks, the optima of the loss functions are connected by curves or volumes, on which the loss is almost constant [48, 52, 37, 21, 70]. Various algorithms have been proposed to find these low-cost curves, which provides a low-cost way to create an ensemble of models from a single trained model. Using our group actions, we propose a new way of constructing models with similar loss values. We show that even with stochasticity in the data, the loss is approximately unchanged under the group action (Appendix 2.9). This provides an efficient alternative to build ensemble models, since the transformation only requires random elements in the symmetry group, without any searching or additional optimization.

We implement our group actions by modifying the activation function between two consecutive layers. Let $H = VX$ be the output of the previous layer. The group action on the weights U, V is

$$g \cdot (U,V) = (U\pi(g,H), gV) \tag{2.54}$$

where $\pi(g,H) = \sigma(H)\sigma(gH)^{\dagger}$. The new activation implements the symmetry group action

$$U\sigma(H) \to U\pi(g,H)\sigma(gH) \tag{2.55}$$

by wrapping the transformations around an activation function $\sigma'(x) = \pi(g,x)\sigma(gx)$, so that $U\sigma'(H) = U\pi(g,H)\sigma(gH)$.

We test the group action on CIFAR-10. The model contains a convolution layer with kernel size 3, followed by a max pooling, a fully connected layer, a leaky ReLU activation, and another fully connected layer. The group action is on the last two fully connected layers. After training a single model, we create transformed models using $g = I + \varepsilon M$, where $M \in \mathbb{R}^{32\times 32}$ is a random matrix and ε controls the magnitude of movement in the parameter space. We then use the mode of the transformed models' prediction as the final output.

We compare the ensemble formed by group actions to four ensembles formed by various random transformation. Let $g = I + \varepsilon M$. The random baselines are:

- 'group': $(U,V) \mapsto (U\pi(g,H), gV)$. This is the model created by group actions.
- 'g^{-1}': $(U,V) \mapsto (Ug^{-1}, gV)$.
- 'random': $(U,V) \mapsto (Ug', gV)$, where $g' = I + \varepsilon D$ and D is a random diagonal matrix.
- 'shuffle': $(U,V) \mapsto (U\pi'(g,H), gV)$, where $\pi'(g,H)$ is constructed by randomly shuffling $\pi(g,H)$.
- 'interpolated permute' or 'perm_interp': $(U,V) \mapsto (U\left(\frac{I+\frac{\varepsilon}{2}(I+S)}{I+\varepsilon}\right)^{-1}, \frac{I+\frac{\varepsilon}{2}(I+S)}{I+\varepsilon}V)$, where $S \in \mathbb{R}^{32\times 32}$ is a random permutation matrix.

Figure 2.8 shows the accuracy of the ensembles compared to single models. The ensemble formed by group actions preserves the model accuracy for small ε and has smaller accuracy drop at larger ε. The ensemble model also improves robustness against Fast Gradient Signed Method (FGSM) attacks (Figure 2.9). Under FGSM attacks with various strength, the ensemble model created using group actions consistently performs better than the baselines with random transformations. However, the same improvement is not observed under Projected Gradient Descent (PGD) attacks.

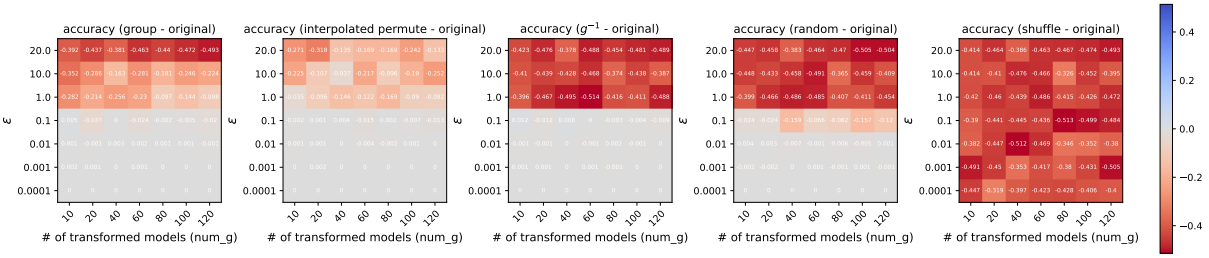
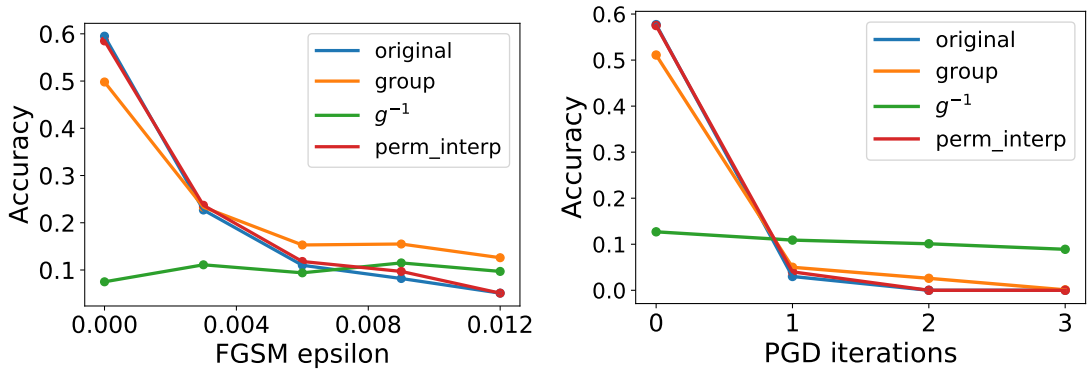


Figure 2.8. Change in accuracy compared to the original single model when using the ensemble model and 4 baselines. The red color indicates degradation in model performance. The ensemble created by group actions has similar loss values when ε is small.



(a) FGSM. (b) PGD.

Figure 2.9. Adversarial attacks on the original model and the ensemble models with various strengths. In FGSM, the group ensemble model improves robustness. In PGD, the ensemble has negligible effects on robustness.

2.10 Discussion

In this paper, we present a general framework of equivariance and introduce a new class of nonlinear, data-dependent symmetries of neural network parameter spaces. These symmetries give rise to conserved quantities in gradient flows, with important implications in improving optimization and robustness of neural networks. While our work sheds new light onto the link between symmetries and group, it contains several limitations, which merit further investigation. First, we have not been able to determine conserved quantities in the radial rescaling case, only a differential equation that gradient flow curves must satisfy. Second, one major contribution of this paper is the non-linear group action of Section 2.4. However, our formulation only gurantees full GL_h equivariance for batch size $k = 1$. In future work, we plan to explore more consequences and variations of this non-linear group action, with the hope of generalizing to greater batch size. Finally, in many cases, parameter space symmetries lead to model compression: i.e., finding a lower-dimension space of parameters with the same expressivity of the original space.

Acknowledgments

This chapter is a full reprint of material published as: Zhao, Bo; Gower, Robert M.; Walters, Robin; Yu, Rose. "Improving Convergence and Generalization Using Parameter

Symmetries." The Twelfth International Conference on Learning Representations (2024). The dissertation author was the primary investigator and author of this paper.

Chapter 3

Understanding Mode Connectivity via Parameter Space Symmetry

In this chapter, we continue to explore symmetry's implication on the loss landscape, focusing on the connectedness of minimum. Neural network minima are often connected by curves along which train and test loss remain nearly constant, a phenomenon known as mode connectivity. While this property has enabled applications such as model merging and fine-tuning, its theoretical explanation remains unclear. We propose a new approach to exploring the connectedness of minima using parameter space symmetry. By linking the topology of symmetry groups to that of the minima, we derive the number of connected components of the minima of linear networks and show that skip connections reduce this number. We then examine when mode connectivity and linear mode connectivity hold or fail, using parameter symmetries which account for a significant part of the minimum. Finally, we provide explicit expressions for connecting curves in the minima induced by symmetry. Using the curvature of these curves, we derive conditions under which linear mode connectivity approximately holds. Our findings highlight the role of continuous symmetries in understanding the neural network loss landscape.

3.1 Introduction

Among recent studies on the loss landscape, a particularly interesting finding is mode connectivity [37, 52]—the observation that distinct minima found by stochastic gradient descent

(SGD) can be connected by continuous, low-loss paths through the high-dimensional parameter space. Mode connectivity has important implications for other aspects of deep learning theory, including the lottery ticket hypothesis [47] and the analysis of loss landscapes and training trajectories [56]. Mode connectivity has also inspired applications in diverse fields, including model ensembling [52, 21, 22], model averaging [70, 145], pruning [47], improving adversarial robustness [163], and fine-tuning for altering prediction mechanism [95].

Despite extensive empirical validation, mode connectivity, especially linear mode connectivity, remains largely a theoretical conjecture [6]. The limited theoretical explanation suggests a need for new proof techniques. In this paper, we focus on parameter symmetries, which encode information about the structure of the parameter space and the minimum. Our work introduces a new approach towards understanding the topology of the minimum and complements existing theories on mode connectivity [153, 48, 109, 110, 81, 127, 111].

Discrete symmetry is known to be related to mode connectivity. In particular, the neural network output, and hence the minimum, is invariant under neuron permutations [63]. Several algorithms have been developed to find optimal permutations for linear connectivity [130, 3], and [41] conjecture that all minima found by SGD are linearly connected up to permutation. Compared to discrete symmetry, the role of continuous symmetry, such as positive rescaling in ReLU, in shaping loss landscape remains less well studied.

We explore the connectedness of minimum through continuous symmetries in the parameter space. Continuous symmetry groups with continuous actions define positive dimensional connected spaces in the minimum [160]. By relating topological properties of symmetry groups to their orbits and the minimum, we show that both continuous and discrete symmetry are useful in understanding the origin and failure cases of mode connectivity. Additionally, continuous symmetry defines curves on the minimum [161]. This enables a principled method for deriving explicit expressions for paths connecting two minima, a task that previously relied on empirical approaches.

Our main contributions are:

- Providing the number of connected components of full-rank linear regression with and without skip connections, by relating topological properties of symmetry groups to those of minima.
- Proving mode connectivity up to permutation for linear networks with invertible weights.
- Deriving examples where the error barrier on linear interpolation of minima is unbounded.
- Deriving explicit low-loss curves that connect minima related by symmetry, and bounding the loss barrier on linear interpolations between minima using the curvature of these curves.

3.1.1 Related Work

Mode connectivity.

[52] and [37] discover empirically that the minima of neural networks are connected by curves on which train and test loss are almost constant. It is then observed that SGD solutions are linearly connected if they are trained from pre-trained weights [108] or share a short period of training at the beginning [47]. Additionally, neuron alignment by permutation improves mode connectivity [130, 136]. Subsequently, [41] conjecture that all minima found by SGD are linearly connected up to permutation. Following the conjecture, [3] develop algorithms that find the optimal alignment for linear mode connectivity, and [72] further reduce the barrier by rescaling the preactivations of interpolated networks.

It is worth noting that linear mode connectivity does not always hold outside of computer vision. Language models that are not linearly connected have different generalization strategies [73]. [95] further show that the lack of linear connectivity indicates that the two models rely on different attributes to make predictions. We derive new theoretical examples of failure cases of linear mode connectivity (Section 3.3.2).

Theory on connectedness of minimum.

Several work explores the theoretical explanation of mode connectivity by studying the connectedness of sub-level sets. [48] show that the minimum is connected for 2-layer linear network without regularization, and for deeper linear networks with $L2$ regularization. Futhermore, they show that the minimum of a two-layer ReLU network is asymptotically connected, that is, there exists a path connecting any two solutions with bounded error. [109] proves that the sublevel sets are connected in pyramidal networks with piecewise linear activation functions and first hidden layer wider than $2N$, where N is the number of training data). The width requirement is later improved to $N+1$ [110].

Others prove connectivity under dropout stability. [81] show that a piece-wise linear path exists between two solutions of ReLU networks, if they are both dropout stable, or both noise stable and sufficiently overparametrized. [127] generalize this proof to show that wider neural networks are more connected, following the observation that SGD solutions for wider network are more dropout stable. [111] give a new upper bound of the loss barrier between solutions using the loss of sparse subnetworks that are optimized, which is a milder condition than dropout stability.

A few papers provide theoretical insights into linear mode connectivity using different approaches. [153] explain linear mode connectivity by finding a convex hull defined by SGD trajectory endpoints. [43] use optimal transport theory to prove that wide two-layer neural networks trained with SGD are linearly connected with high probability. [129] explain the topography of the loss landscape that enables or obstructs linear mode connectivity. [167] show that the feature maps of each layer are also linearly connected and identify conditions that guarantee linear connectivity. [6] analyze effects of architecture, optimization algorithm, and dataset on linear mode connectivity empirically.

We approach the theoretical origin of mode connectivity via continuous symmetries in the parameter space, a connection that has not been previously established. This connection leads to new topological results and explicit expressions of low loss curves. Using these results,

we also contribute to the understanding for linear mode connectivity by providing conditions under which it approximately holds.

Symmetry in the loss landscape.

Discrete symmetries have inspired a line of work on loss landscapes. [24] show that permutations of a layer are connected within a loss level set. By analyzing permutation symmetries, [128] characterize the geometry of the global minima manifold for networks and show that adding one neuron to each layer in a minimal network connects the permutation equivalent global minima. Continuous symmetries have also gained attention in optimization [16, 114, 82, 157]. By removing permutation and rescaling symmetries, [115] study the geometry of minima in the functional space. [160] find a set of nonlinear continuous symmetries that partially parametrizes the minimum. [161] use symmetry induced curves to approximate the curvature of the minimum. Our paper explores a new application of parameter symmetry—explaining the connectedness of the minimum.

3.1.2 Preliminaries

Consider two topological spaces X and Y. A map $f : X \to Y$ is *continuous* if for every open subset $U \subseteq Y$, its preimage $f^{-1}(U)$ is open in X. If X and Y are metric spaces with metrics d_X and d_Y respectively, this is equivalent to the delta-epsilon definition. That is, f is continuous if at every $x \in X$, for any $\varepsilon > 0$ there exists $\delta > 0$ such that $d_X(x,y) < \delta$ implies $d_Y(f(x), f(y)) < \varepsilon$ for all $y \in X$.

A topological space is *connected* if it cannot be expressed as the union of two disjoint, nonempty, open subsets. A topological space X is *path connected* if for every $p, q \in X$, there is a continuous map $f : [0,1] \to X$ such that $f(0) = p$ and $f(1) = q$. Path connectedness implies connectedness. The converse is not always true [89], but counterexamples are often specifically constructed and unlikely to be encountered in the context of deep learning. Path connectedness can therefore help develop intuition for connectedness, for practical purposes.

The following theorem is the main intuition of this paper and will appear frequently in proofs.

Theorem 3.1.1 (Theorem 4.7 in [89])**.** *Let X, Y be topological spaces and let $f : X \to Y$ be a continuous map. If X is connected, then $f(X)$ is connected.*

A map f is a *homeomorphism* from X to Y if f is bijective and both f and f^{-1} are continuous. X and Y are *homeomorphic* if such a map exists. A *(connected) component* of a topological space X is a maximal nonempty connected subset of X. The components of X form a partition of X. The next two corollaries of Theorem 3.1.1 show that connectedness and the number of connected components are topological properties. That is, they are preserved under homeomorphisms.

Corollary 3.1.2. *Let $f : X \to Y$ be a homeomorphism from X to Y, and let $U \subseteq X$ be a subset of X with the subspace topology. Then U is connected if and only if $f(U) \subseteq Y$ is connected.*

Corollary 3.1.3. *Let X be a topological space that has N components. Let Y be a topological space homeomorphic to X. Then Y has N components.*

3.2 Connected Components of the Minimum

In this section, we relate topological properties of symmetry groups to topological properties of the minimum. In particular, we provide the number of connected components of the minimum when all symmetries are known. Omitted proofs can be found in Appendix B.1.

3.2.1 Linear Network with Invertible Weights

Let Param be the space of parameters. Consider the multi-layer loss function $L : \mathsf{Param} \to \mathbb{R}$,

$$L : \mathsf{Param} \to \mathbb{R}, \qquad (W_1, ..., W_l) \mapsto ||Y - W_l ... W_1 X||_2^2. \tag{3.1}$$

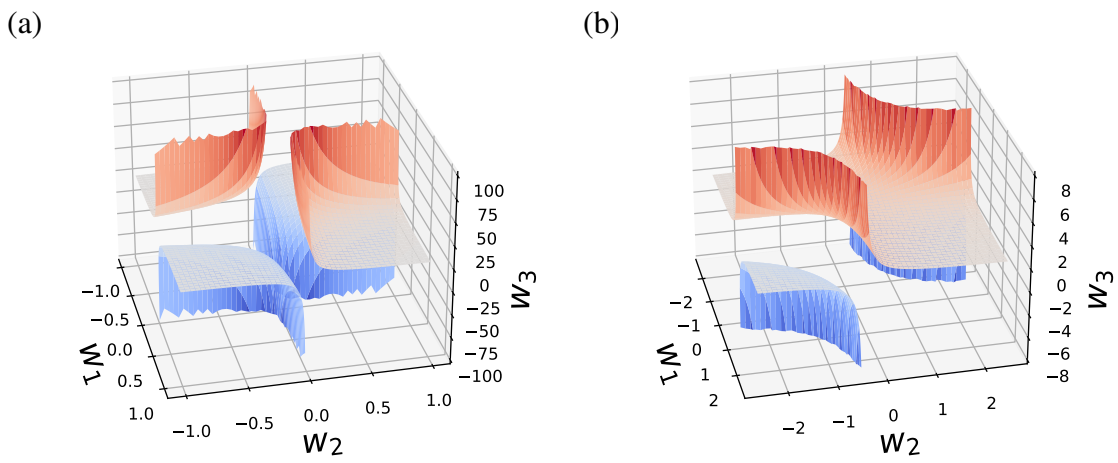


Figure 3.1. Minimum of (a) a 3-layer linear net $||Y - W_3W_2W_1X||_2$ and (b) a 3-layer linear net with a residual connection $||Y - W_3(W_2W_1X + X)||_2$, where $X = 1$, $Y = 1$, and $W_1, W_2, W_3 \in \mathbb{R}$.

where $X, Y \in \mathbb{R}^{h\times h}$ are the input and output of the network. In this subsection, we assume that both X, Y have rank h, and $\text{Param} = (\mathbb{R}^{h\times h})^l$. Then L is invariant to $GL_h(\mathbb{R})^{l-1}$, which acts on Param by $g \cdot (W_1, ..., W_l) = (g_1W_1, g_2W_2g_1^{-1}, ..., g_{l-1}W_{l-1}g_{l-2}^{-1}, W_lg_{l-1}^{-1})$, for $(g_1, ..., g_{l-1}) \in GL_h(\mathbb{R})^{l-1}$.

Let $L^{-1}(c) = \{\theta \in \text{Param} : L(\theta) = c\}$ be a level set of L. Since $\|\cdot\|_2 \geq 0$ and $L^{-1}(0) \neq \emptyset$, the minimum value of L is 0. By relating the topology of $GL_h(\mathbb{R})$ and $L^{-1}(0)$, we have the following observations on the structure of the minimum of L.

Proposition 3.2.1. *There is a homeomorphism between* $L^{-1}(0)$ *and* $(\text{GL}_h)^{l-1}$.

Since $(\text{GL}_h)^{l-1}$ has 2^{l-1} connected components and homeomorphisms preserve topological properties, $L^{-1}(0)$ also has 2^{l-1} connected components. Note that this number is independent of the network width, due to the fact that $GL_n(\mathbb{R}))$ has two connected components regardless of n.

Corollary 3.2.2. *The minimum of L has* 2^{l-1} *connected components.*

3.2.2 ResNet with 1D Weights

The topological properties of the minimum set depend on the architecture. As an example of this dependency, we show that adding a skip connection changes the number of connected components of the minimum.

Consider a residual network $W_3(W_2W_1X + \varepsilon X)$ and loss function

$$L(W_3, W_2, W_1) = ||Y - W_3(W_2W_1X + \varepsilon X)||_2, \tag{3.2}$$

where $(W_1, W_2, W_3) \in \mathsf{Param} = \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$, $\varepsilon \in \mathbb{R}$, and data $X \in \mathbb{R}^{n\times n}, Y \in R^{n\times n}$. The following proposition states that for a three-layer residual network with weight matrices of dimension 1×1, the number of components of the minimum is smaller than that of a linear network without the skip connection.

Proposition 3.2.3. *Let $n = 1$. Assume that $X, Y \neq 0$. When $\varepsilon = 0$, the minimum of L has 4 connected components. When $\varepsilon \neq 0$, the minimum of L has 3 connected components.*

The $\varepsilon = 0$ case follows from Corollary 3.2.2. For the $\varepsilon \neq 0$ case, the proof decomposes the minimum of L into two sets S_1 and S_0, corresponding to the minima without the skip connection and an extra set of solutions because of the skip connection. S_1 is homeomorphic to $GL_1 \times GL_1$ and has 4 connected components. S_0 is a line and has 1 connected component. Two components of S_1 are connected to S_0, while the other two components of S_1 are not. Therefore, S_0 connects two components of S_1. As a result, the minimum of L has 3 connected components.

Figure 3.1 visualizes the minimum without and with the skip connection. This result reveals the effect of skip connection on the connectedness of the set of minima, which may lead to a new explanation of the effectiveness of ResNets [62] and DenseNets [66]. We leave the connection between the topology of the minimum and the optimization and generalization properties of neural networks to future work.

3.3 Mode Connectivity

The previous section counts the connected components of the minimum and shows that the connectedness of the minimum is related to the symmetry of the loss function under certain conditions. In this section, we use this insight to explain recent empirical observations that with high probability two points in the minimum are connected, i.e. there is a large connected component. Proofs of this section appears in Appendix B.2.

Mode connectivity refers to the phenomenon that there exist high accuracy or low loss paths between two minima found by stochastic gradient descent [52]. Linear mode connectivity occurs when all points on the linear interpolation between two minima have low loss values. More recently, permutation of neurons is usually performed to align the two minima before evaluating linear mode connectivity [41, 3]. We use the term mode connectivity when we consider arbitrary curves and will specify linear mode connectivity when only linear interpolation is considered.

3.3.1 Mode Connectivity up to Permutation

For the family of linear neural networks defined in Section 3.2.1, we show that permutations allow us to connect points in the minimum that are not connected without permutation. Our results support the empirical observation that neuron alignment by permutation improves mode connectivity [136].

Consider again the linear network equation 3.1 with invertible weights. When $l = 2$, the minimum of L has two connected components corresponding to the two connected components of the GL group. Any $g \in GL$ that is not on the identity component can take a point on one connected component of the minimum to the other.

Lemma 3.3.1. *Consider two points* $(W_1, W_2), (W_1', W_2') \in L^{-1}(0)$ *that are not connected in* $L^{-1}(0)$. *For any* $g \in GL(h)$ *such that* $\det(g) < 0$, $g \cdot (W_1, W_2)$ *and* (W_1', W_2') *are connected in* $L^{-1}(0)$.

When the hidden dimension $h \geq 2$, there exists a permutation g such that $\det(g) > 0$, and a permutation g such that $\det(g) < 0$. Therefore, Lemma 3.3.1 implies the following result that

all points on the minimum of L are connected up to permutation.

Proposition 3.3.2. *Assume that $h \geq 2$. For all $(W_1,...,W_l),(W_1',...,W_l') \in L^{-1}(0)$, there exists a list of permutation matrices $P_1,...,P_{l-1}$ such that $(W_1P_1, P_1^{-1}W_2P_2,...,P_{l-2}W_{l-1}P_{l-1}, P_{l-1}W_l)$ and $(W_1',...,W_l')$ are connected in $L^{-1}(0)$.*

The results above are examples where a larger part of the minimum becomes connected after a permutation. More generally, permutation improves mode connectivity in cases where an orbit is not connected due to the symmetry group comprising multiple connected components, the orbit does not reside on the same connected component of the minimum, and there exists a permutation that takes a point on one connected component of the group to another.

3.3.2 Failure Case of Linear Mode Connectivity

As an application of obtaining new minima from old ones using symmetries, we show that linear mode connectivity fails to hold in multi-layer regressions. The following proposition says that in neural networks with a homogeneous activation (such as leaky ReLU) between the last two layers, the error barrier in the linear interpolation between two solutions can be arbitrarily large.

Proposition 3.3.3. *Consider a loss function of the following form*

$$\begin{aligned} &L : \mathsf{Param} \to \mathbb{R}, W = (W_1,...,W_l) \mapsto \\ &\quad ||Y - W_l\sigma(W_{l-1}f(W_{l-2},W_{l-3},...,W_1,X))||_2^2, \end{aligned} \tag{3.3}$$

where f is a function of $W_{l-2},W_{l-3},...,W_1,X$, and $\sigma(cz) = c^k\sigma(z)$ for all $c \in \mathbb{R}$ and some $k > 0$. Assume that $||Y||_2 \neq 0$ and $L^{-1}(0) \neq \emptyset$. Also assume that $l \geq 2$. For any positive number $b > 0$, there exist $W, W' \in L^{-1}(0)$ that belong to the same connected component of $L^{-1}(0)$ and $0 < \alpha < 1$, such that $L((1-\alpha)W + \alpha W') > b$.

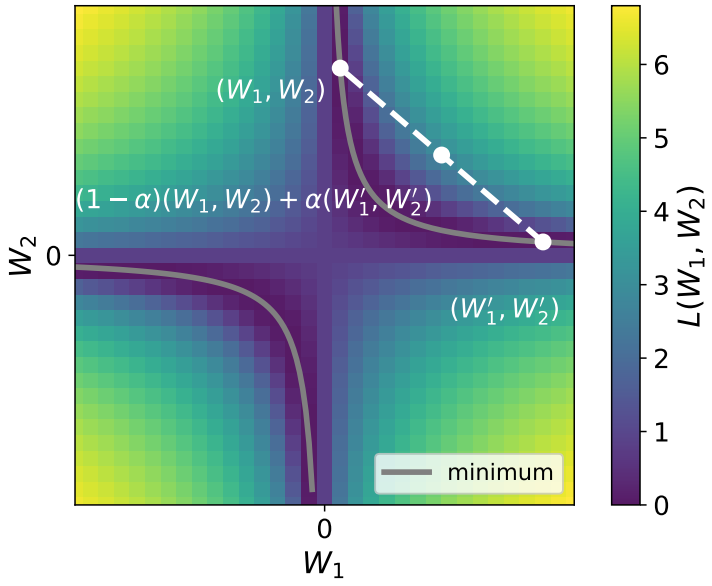


Figure 3.2. Interpolation between 2 minima of loss function $L(W_1, W_2) = ||Y - W_2 W_1 X||_2$ with 1 dimensional weights. Loss on the interpolation can be unbounded.

The proof constructs a new point on the minimum from an existing one using the rescaling symmetry of homogeneous functions. The two points can be far apart since the orbit of this group action is unbounded. To provide intuition, Figure 3.2 visualizes the two points on the minimum of a two-layer network with weights of dimension 1×1 and the linear interpolation between them. The linear network used is a special case of a homogeneous network. Note that our result here does not contradict with the layer-wise connectivity result in [1], as more than one layer of the two minima are different.

The loss function considered in Proposition 3.3.3 is significantly more general than those in Section 3.3.1. For the architecture, we only require the presence of a rescaling symmetry in the last two layers, and f can be any neural network with any activation. Other assumptions of the proposition are also not excessively restrictive, as the labels Y are rarely all zero, and there usually exists a minimum in common machine learning tasks.

Proposition 3.3.3 extends to cases where we allow certain permutations. The following proposition states that under additional assumptions, the error barrier in the linear interpolation is unbounded even with neuron permutations. The proof construction is similar to that of

Proposition 3.3.3.

Let S_n be the set of $n \times n$ permutation matrices, where n is the number of columns of W_l.

Proposition 3.3.4. *Consider the loss function with the same set of assumptions in Proposition 3.3.3. Assume additionally that there does not exist a permutation P such that every column of $P\sigma(W_{l-1}f(W_{l-2}, W_{l-3}, ..., W_1, X))$ is in the null space of W_l. For any positive number $b > 0$, there exist $(W_1, ..., W_l), (W'_1, ..., W'_l) \in L^{-1}(0)$ and $0 < \alpha < 1$, such that $(W_1, ..., W_{l-2}) = (W'_1, ..., W'_{l-2})$ and*

$$\min_{P \in S_n} L\big((1-\alpha)(W_1, ..., W_l) + \alpha(W_1, ..., W_{l-2}, P^{-1}W_{l-1}, W_l P)\big) > b.$$

By including permutation, the setting in Proposition 3.3.4 is closer to the setting in which linear mode connectivity is empirically observed. However, the permutation in Proposition 3.3.4 is restricted to the first two layers, which does not rule out the possibility of lowering the loss barrier by including permutations of other neurons.

The proofs of Proposition 3.3.3 and 3.3.4 depend on the rescaling symmetry of homogenenous activation functions. For other activations with known symmetries, similar results may be derived as using the large set of minimum obtained from the group action. Whether the loss barrier on the linear interpolation is bounded can depend on the compactness of the symmetry group and the curvature of the minimum. We leave a systematic investigation of the condition for linear mode connectivity to future work.

One possible reason why linear mode connectivity is observed in practice despite Proposition 3.3.4 is that only a small part of the minima is reachable by stochastic gradient descent due to implicit bias [101], as other optimizers have been observed to find less connected minima [6].

3.3.3 Linear Mode Connectivity of Orbits

Symmetry accounts for a large part of the set of minima. In particular, given a known minimum x, the orbit of x defines a set of points that are also minima. Although not all minima are on the same orbit of known symmetries, each orbit often contains a nontrivial set of minima. In this section, we examine the error barrier of linear interpolations of minima restricted to an orbit of parameter symmetries.

When the architecture contains a multiplication of two weight matrices W_2W_1, where $W_2 \in \mathbb{R}^{m\times h}, W_1 \in \mathbb{R}^{h\times n}$, there is a GL_h symmetry that acts on (W_1, W_2) by $g \cdot (W_1, W_2) = (gW_1, W_2g^{-1})$ for $g \in GL_h$. The following proposition states that a point on the linear interpolation of two points in the same orbit can be far away from the orbit.

Proposition 3.3.5. *Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix. Let set $S = \{(W_1, W_2) : W_1, W_2 \in \mathbb{R}^{n\times n}, W_1W_2 = A\}$. For any positive number $b > 0$, there exist $W', W'' \in S$ and $0 < \alpha < 1$, such that $\min_{\hat{W}\in S} \| ((1-\alpha)W' + \alpha W'') - \hat{W} \|_2 > b$.*

The structure in the form of W_1W_2 is not uncommon in deep learning architectures. Notably, the parameter matrices for queries and keys in the attention function are multiplied directly in this manner [140], thus admitting the GL_h symmetry and having orbits with properties given by Proposition 3.3.5.

While the error barrier in the linear interpolation of two minima can be unbounded (Proposition 3.3.3), this typically occurs when the parameters are allowed to be arbitrarily large. Constraining the parameters to remain bounded ensures that the loss barrier is bounded above. The following proposition makes this intuition precise for the set of minima consisting of a particular orbit.

Proposition 3.3.6. *Consider the loss function with the same set of assumptions in Proposition 3.3.3. Let $W \in L^{-1}(0)$ be a point on the minimum. Consider the multiplicative group of positive real numbers $\mathbb{R}^+$ that acts on $L^{-1}(0)$ by $g \cdot (W_1, ..., W_l) = (W_1, ..., W_{l-2}, gW_{l-1}, W_l g^{-k})$, where*

$g \in \mathbb{R}^+$. *Then there exists a positive number* $b > 0$, *such that for all* $0 < \alpha < 1$ *and* $W' \in Orbit(W)$ *with* $||W_i'||_2 < c$ *for all* i *and some* $c > 0$, *the loss value for points on the linear interpolation* $L((1-\alpha)W + \alpha W') < b$.

Proposition 3.3.5 and 3.3.6 are two examples where the knowledge of parameter symmetry enables analysis of the linear connectivity of subsets of minima. As more continuous symmetries are characterized (e.g. the nonlinear symmetries in [160]), these analysis can potentially be extended to even larger parts of the set of minima.

3.4 Curves on Minimum from Group Actions

The paths connecting two points in the set of minima may not be linear. Previously, these paths were discovered empirically by finding parametric curves on which the expected loss is minimized [52]. Using parameter space symmetry, we uncover an alternative and principled way to find curves on the minimum.

3.4.1 Symmetry Induced Curves

Suppose the loss function $L : \mathsf{Param} \to \mathbb{R}$ is invariant with respect to some Lie group G. Consider the following curve for a point $\boldsymbol{w} \in \mathsf{Param}$ and $M \in \mathrm{Lie}(G)$:

$$\begin{aligned} \gamma_M &: \mathbb{R} \times \mathsf{Param} \to \mathsf{Param}, \\ \gamma_M(t, \boldsymbol{w}) &= \exp(tM) \cdot \boldsymbol{w}. \end{aligned} \tag{3.4}$$

Since $\exp(tM) \in G$ and the action of G preserves the value of L, every point on γ_M is in the same L level set as $\boldsymbol{w}$. This provides a way to find a curve of constant loss between two points that are in the same orbit. Concretely, given two points $\boldsymbol{w}_1$ and $\boldsymbol{w}_2 = g \cdot \boldsymbol{w}_1$, let γ be the following curve:

$$\begin{aligned} \gamma &: [0,1] \times G \times \mathsf{Param} \to \mathsf{Param}, \\ \gamma(t, g, \boldsymbol{w}) &= \exp(t \log(g)) \cdot \boldsymbol{w}. \end{aligned} \tag{3.5}$$

Note that $\gamma(0,g,\boldsymbol{w}_1)=\boldsymbol{w}_1$, $\gamma(1,g,\boldsymbol{w}_1)=\boldsymbol{w}_2$, and $L(\gamma(t,g,\boldsymbol{w}_1))=L(\boldsymbol{w}_1)=L(\boldsymbol{w}_2)$ for all $t\in[0,1]$. Hence, γ is a curve that connects the points $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, and every point on γ has the same loss value as $L(\boldsymbol{w}_1)=L(\boldsymbol{w}_2)$.

For a group G, the curve γ is defined when the map $\cdot : G\times \mathsf{Param}\to\mathsf{Param}$ is continuous and $\mathrm{id}\cdot\boldsymbol{w}=\boldsymbol{w}$ for all $\boldsymbol{w}\in\mathsf{Param}$, even if it is not a group action or does not preserve loss. However, when $\cdot$ does not preserve loss, the loss can change on γ. Consider our two-layer network and the following map:

$$\begin{aligned}\cdot : GL(h,\mathbb{R})\times\mathsf{Param}&\to\mathsf{Param}\\ g\cdot(U,V)&=(U\sigma(VX)\sigma(gVX)^{\dagger},gV).\end{aligned}\tag{3.6}$$

When σ is the identity function, $\cdot$ preserves the loss value, and γ defines a curve on the minimum. In general, the map *equation* 3.6 does not preserve loss when batch size k is larger than hidden dimension h. However, the maximum change of loss on γ can be bounded as follows.

Proposition 3.4.1. *Let* $(U,V)\in\mathsf{Param}$*, and* $(U',V')=g\cdot(U,V)$*. Then*

$$\|U\sigma(VX)-U'\sigma(V'X)\|\leq\|U\sigma(VX)\|.\tag{3.7}$$

We demonstrate Proposition 3.4.1 empirically using a set of two-layer networks with various parameter space dimensions. Specifically, we construct networks in the form of $\|U\sigma(VX)-Y\|^2$, with σ being the sigmoid function, $X\in\mathbb{R}^{n\times k}, Y\in\mathbb{R}^{m\times k}$, and $(U,V)\in\mathsf{Param}=\mathbb{R}^{m\times h}\times\mathbb{R}^{h\times n}$. We create 100 such networks, each with m,h,n,k randomly sampled from integers between 2 and 100. In each network, elements in X and Y are sampled independently from a normal distribution, and U,V are randomly initialized. After training with SGD, we compute $(U',V')=g\cdot(U,V)$ using equation 3.6 with a random invertible matrix g. We then plot $\|U\sigma(VX)\|$ against $\|U\sigma(VX)-U'\sigma(V'X)\|$ in Figure 3.3(a). All points are above the line $y=x$, as predicted by Proposition 3.4.1.

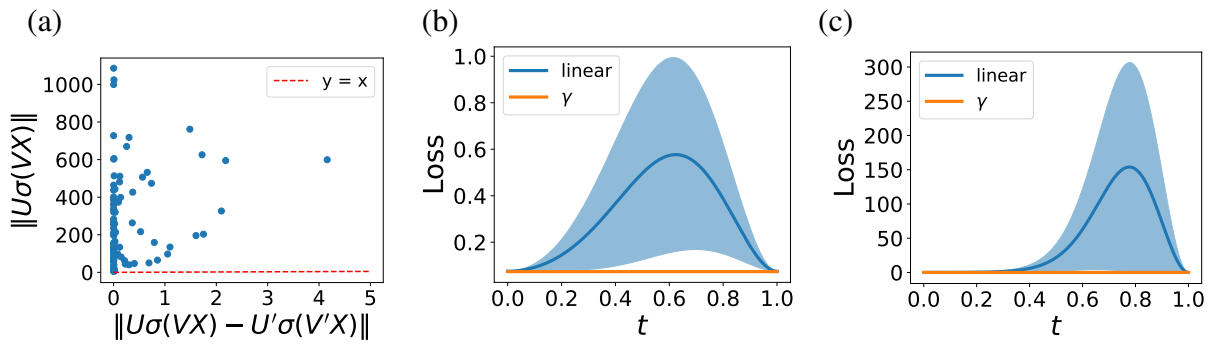


Figure 3.3. (a) Empirical validation of Proposition 3.4.1. (b-c) The loss on the curves induced by approximate symmetries (γ) remains relatively low, compared to the loss on the linear interpolation between the two ends of these curves. (b) and (c) differ by the magnitude of the group element used. The loss is averaged over 5 random curves.

While the map equation 3.6 is not a group action in general, it connects more points in the set of minima than only using known symmetries, and the points on the connecting curves have bounded loss. Figure 3.3(b-c) shows that the loss on the curves induced by approximate symmetries remains relatively low, compared to the loss on the linear interpolation between the two ends of these curves. We consider a two layer network with loss function $\|W_2\sigma(W_1X) - Y\|$, with σ being a leaky ReLU function, $X \in \mathbb{R}^{16\times 8}, Y \in \mathbb{R}^{64\times 8}$, and $(W_1, W_2) \in \text{Param} = \mathbb{R}^{32\times 16} \times \mathbb{R}^{32}$. In the figures, γ denotes a curve obtained using Equation equation 3.5 together with equation 3.6. The starting point of γ is a minimum found by SGD. Both γ and the linear interpolation are parametrized by $t \in [0,1]$. Compared to the linear interpolation between the two end points of γ, the loss on γ is consistently lower. Figure 3.3(c) uses group elements with larger magnitudes, resulting in a larger distance between $\gamma(0)$ and $\gamma(1)$, which might explain the higher loss barrier on their linear interpolation.

3.4.2 Approximate Linear Connectivity under Bounded Curvature of Minima

Knowing the explicit expression of connecting curves brings new insight into when linear mode connectivity approximately holds. In particular, these expressions provide information

about the curvature of the curves. If the curvatures are small, then there exists an approximately straight line connecting any two minima along which the loss remains close to its minimum value.

Consider a loss level set $L^{-1}(c) = \{\boldsymbol{w} \in \mathsf{Param} : L(\boldsymbol{w}) = c\}$ with some $c \in \mathbb{R}$. Suppose we have two points $\boldsymbol{w}_1, \boldsymbol{w}_2 \in L^{-1}(c)$ connected by a smooth curve γ lying entirely within $L^{-1}(c)$. The curvature of γ can be written as $\kappa(\gamma, t) = \frac{\|T'(t)\|}{\|\gamma'(t)\|}$, where $\gamma' = \frac{d\gamma}{dt}$ and $T(t) = \frac{\gamma'(t)}{\|\gamma'(t)\|}$. If the curvature of this curve is small or bounded, we can show that there exists an approximately straight line connecting $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ that remains close to $L^{-1}(c)$. Additionally, if L is Lipschitz continuous, its value remains close to c along this line segment. We formalize this with the following theorem.

Theorem 3.4.2. *Let $L^{-1}(c) \subset \mathsf{Param}$, with $c \in \mathbb{R}$, be a level set of the loss function $L : \mathsf{Param} \to \mathbb{R}$. Let $\gamma : [0,1] \to L^{-1}(c)$ be a smooth curve in $L^{-1}(c)$ connecting two points $\boldsymbol{w}_1 = \gamma(0)$ and $\boldsymbol{w}_2 = \gamma(1)$. Suppose the curvature $\kappa(t)$ of γ satisfies $\kappa(t) \leq \kappa_{\max}$ for all $t \in [0,1]$.*

Let S be the straight line segment connecting $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Then, for any point $\boldsymbol{w}$ on S, the distance to $L^{-1}(c)$ is bounded by

$$\operatorname{dist}(\boldsymbol{w}, L^{-1}(c)) \leq d_{\max}, \tag{3.8}$$

with

$$d_{\max} = \frac{1}{\kappa_{\max}} \left(1 - \sqrt{1 - \left(\frac{\kappa_{\max} \|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2}{2} \right)^2} \right).$$

Furthermore, assuming L is Lipschitz continuous with Lipschitz constant C_L, the loss at any point $\boldsymbol{w}$ on S satisfies

$$|L(\boldsymbol{w}) - c| \leq C_L d_{\max}. \tag{3.9}$$

When the group action induces curves with bounded curvature, Theorem 3.4.2 applies.

Since the minimum is also a level set of L, Theorem 3.4.2 provides a sufficient condition for linear mode connectivity to approximately hold. When the curvature of the minimum is small, points on the minimum are approximately connected through nearly straight paths along with the loss does not increase significantly. If $\kappa_{\max}\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|$ is small, we can use the first-order approximation of the square root and obtain $d_{\max} \approx \frac{\kappa_{\max}\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2^2}{8}$.

3.5 Discussion

In this chapter, we saw that topological properties of symmetry groups are useful to infer topological properties of the loss level sets. Specifically, we derive the number of connected components of full-rank multi-layer networks with and without skip connections, and prove mode connectivity up to permutation for full-rank linear regressions. Using symmetry in the parameter space, we construct an explicit expression for curves that connect two points in the same orbit. The explicit expressions allow us to obtain the curvature of these curves, which are useful to bound the loss barrier on linear interpolation between minima.

For practitioners, these results motivate concrete strategies—and cautions—for tasks that navigate the loss landscape, including model merging, ensembling, and fine-tuning:

- One can build low-loss curves explicitly using known parameter symmetries. This gives a principled and efficient way to obtain new minima from old ones, potentially useful for (1) generating model ensembles with low cost; (2) improving model alignment by allowing a much larger group of transformations than permutation; and (3) mitigating catastrophic forgetting in fine-tuning by constraining updates to the symmetry-induced manifold of the pretraining minimum.

- The connectedness of minima supports the practice of model merging and ensembling, even when models are trained separately. In addition to permutation, many other symmetry transformations can connect solutions that would otherwise appear very different.

- Linear interpolation between minima is not guaranteed to lead to better models, despite its widespread use. This highlights the need to evaluate whether the minima found by specific learning algorithms are approximately connected before averaging models directly.

Acknowledgments

This chapter is a full reprint of material published as: Zhao, Bo; Dehmamy, Nima; Walters, Robin; Yu, Rose. "Understanding Mode Connectivity via Parameter Space Symmetry." In International Conference on Machine Learning, pp. 77441-77460. PMLR 2025. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Symmetry Teleportation I: Algorithm, Theory, and Empirical Results

In this chapter, we study an application of symmetry in optimization. Existing gradient-based optimization methods update parameters locally, in a direction that minimizes the loss function. We study a different approach, symmetry teleportation, that allows parameters to travel a large distance on the loss level set, in order to improve the convergence speed in subsequent steps. Teleportation exploits symmetries in the loss landscape of optimization problems. We prove a necessary condition for teleportation to improve convergence rate, and show that teleportation not only speeds up optimization in the short-term, but gives overall faster time to convergence. We then show that our algorithm is closely related to second order methods. We also explore conditions under which one teleportation is enough to guarantee an optimal trajectory. Experimentally, we show that teleportation improves the convergence speed of gradient descent for several optimization problems, including test functions, multi-layer regressions, and MNIST classification.

4.1 Introduction

Consider the optimization problem of finding $\text{argmin}_{\boldsymbol{w}} L(\boldsymbol{w})$, where L is a loss function and $\boldsymbol{w}$ are the parameters. While finding global minima of $L(\boldsymbol{w})$ is hard for non-convex problems, we can use gradient descent (GD) to find local minima. In GD we apply the following update

rule at every step:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \nabla L, \tag{4.1}$$

where η is the learning rate. Gradient descent is a first-order method that uses only gradient information. It is easy to compute but suffers from slow convergence. Second-order methods such as Newton's method use the second derivative to account for the landscape geometry. These methods enjoy faster convergence, but calculating the second derivative (Hessian) can be computationally expensive over high dimensional spaces [61].

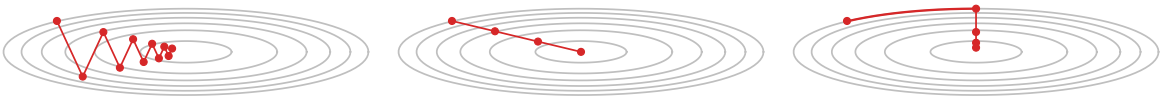


Figure 4.1. Left to right: gradient descent, second-order methods, gradient descent after a teleportation.

In this paper, we propose a new optimization method based on parameter space symmetries, which are group actions on the parameter space that leave the loss unchanged. Our algorithm takes advantage of higher-order landscape geometry but uses only gradient information in most steps, thus avoiding the computational cost of second-order methods.

Specifically, we look beyond local optimization and ask: *what if we allow the parameters to make a big jump once in a while?* As shown in Figure 4.1, during optimization, we teleport the parameters to another point with steeper gradients using a loss-invariant symmetry transformation. After teleportation, the loss stays the same, but the gradient and hence the rate of loss decay changes. The increased magnitude of the gradient can reduce the number of steps required to converge, leading to acceleration of the gradient descent.

The locality of gradient descent is reflected in its formulation in terms of a proximal term; our method circumvents this locality by teleporting to new locations with the same loss. A step of GD is equivalent to the following proximal mapping [30]. Let $\langle \cdot, \cdot \rangle$ denote the inner product,

we have

$$\boldsymbol{w}_{t+1} = \text{argmin}_{\boldsymbol{w}} \left\{ \eta \langle (\nabla L)|_{\boldsymbol{w}_t}, \boldsymbol{w} \rangle + \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|_2^2 \right\}. \tag{4.2}$$

The term $\frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2$ is the proximal term that keeps $\boldsymbol{w}_{t+1}$ close to $\boldsymbol{w}_t$. Adaptive gradient methods define the proximal term using the Mahalanobis distance $\|\boldsymbol{w} - \boldsymbol{w}_t\|^2_{G^{-1/2}}$ to account for landscape geometry. For example, in AdaGrad, G is the sum of the outer product of gradients [39]. Our proposed teleportation technique relaxes the requirement from the proximal term. We teleport to points on the same level set of $L(\boldsymbol{w}_t)$, but allow $\boldsymbol{w}$ to be far from $\boldsymbol{w}_t$ in Euclidean distance.

4.2 Related Work

Continuous parameter space symmetries have been identified in neural networks with homogeneous [16, 38] and radial activation functions [51]. The effect of symmetry transformations on gradients has been examined for translation, scale, and rescale symmetries in [82]. We contribute to this line of work by deriving loss-invariant group actions in multi-layer neural networks with invertible activation functions.

Several works exploit symmetry to facilitate optimization. For example, Path-SGD [107] improves optimization in ReLU networks by path regularization. $\mathscr{G}$-SGD [99] performs weight updates in a scale-invariant space, which also exploits the symmetry in ReLU networks. [139] analyzes the effect of L2 regularization in batch normalization which gives scale-invariant functions. [17] transforms parameters in the symmetry orbits to address slow movement along directions of weakly broken symmetry and find the optimal $g \in G$ that minimizes $L(g \cdot \boldsymbol{w})$. In comparison, we search within G-orbits for points which maximize $|dL(g \cdot \boldsymbol{w})/dt| = \|\nabla L(g \cdot \boldsymbol{w})\|^2$.

Natural gradient [7], adaptive gradient methods [39, 78], and their approximations [96, 60] improve the direction of parameter updates instead of directly transforming parameters. If the group acts transitively on the level set, and we teleport to the point with maximum gradient,

then our update direction is the same as that in Newton's method. We prove that our algorithm is connected to second-order optimization methods and show that teleportation can be used to improve these methods empirically.

The concept of neural teleportation was first explored using quiver representation theory [10, 11]. These works provide a way to explore level curves of the loss of neural networks and show that random teleportation speeds up gradient descent experimentally and theoretically. Our algorithm improves neural teleportation by searching for teleportation destinations that lead to the largest improvement in the magnitude of gradient.

Several works study how parameter initialization affects convergence rate [123, 135, 101]. If we apply a group transformation only at initialization, our method is similar to that of [135]. We do not guarantee that the transformed parameters lead to faster convergence rate throughout the entire training. However, we accelerate convergence at least for a short time after initialization. Additionally, our method is not restricted to initialization and can be applied at any time during training.

Most contemporary neural networks are overparametrized. While this has been shown to improve generalization [19], an important question is how overparametrization affects optimization. A series of works starting from [13] show that overparametrization resulting from the depth of a neural network accelerates convergence. Another view is that the symmetry created by overparametrization poses constraints on trajectories in the form of conserved quantities [54]. Additionally, the symmetry generates additional trajectories. When the trajectories created by overparametrization are equivalent, model compression by removing symmetry reduces training time [51]. However, when trajectories are not equivalent, gradient flows on some paths are faster than others. We search for the better paths created by overparametrization.

4.3 Symmetry Teleportation

We propose *symmetry teleportation*, an accelerated gradient-based optimization algorithm that exploits symmetries in the loss landscape. Symmetry teleportation searches for the best gradient descent trajectory by teleporting parameters to a point with larger gradients using a group action. The resulting algorithm requires only gradient computations but is able to account for the global landscape geometry, leading to faster loss decay.

Let the group G be a set of symmetries which leave the loss function L invariant: $L(g \cdot (\boldsymbol{w}, X)) = L(\boldsymbol{w}, X)$, where $g \in G$. We perform gradient descent for t_{max} steps. We define an index set $K \subseteq \{0, 1, 2, ..., t_{max} - 1\}$ as a teleportation schedule. At epochs that are in the schedule, we transform parameters using group element $g \in G$ to the location where the gradient is largest, then continue with gradient descent. Algorithm 1 describes the details of this procedure. Note that the loss does not change after teleportation (Line 2-5) since L is G-invariant.

Algorithm 1: Symmetry Teleportation

Input: Loss function $L(\boldsymbol{w})$, learning rate η, number of epochs t_{max}, initialized parameters $\boldsymbol{w}_0$, symmetry group G, teleportation schedule K.
Output: $\boldsymbol{w}_{t_{max}}$.

1 **for** $t \leftarrow 0$ **to** $t_{max} - 1$ **do**
2 **if** $t \in K$ **then**
3 $g \leftarrow \text{argmax}_{g \in G} \|(\nabla L)|_{g \cdot \boldsymbol{w}_t}\|^2$
4 $\boldsymbol{w}_t \leftarrow g \cdot \boldsymbol{w}_t$
5 **end if**
6 $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta (\nabla L)|_{\boldsymbol{w}_t}$
7 **end for**
8 **return** $\boldsymbol{w}_{t_{max}}$

Algorithm 1 can be generalized to apply teleportation to stochastic gradient descent. We discuss some design choices below and provide detailed analysis in the next two sections.

When the action of G is continuous, teleportation can be implemented by parameterizing and performing gradient ascent on g. For example, the $\text{SO}(2)$ group can be parameterized by the rotation angle θ. Small transformations $g \in \text{GL}_d(\mathbb{R})$ (general linear group) can be parameterized

as $g \approx I + \varepsilon T$ where $\varepsilon \ll 1$ and T are arbitrary $d \times d$ matrices. For discrete groups, search algorithms or random sampling can be used to find a group element that improves the magnitude of the gradient.

Although $g \cdot (\boldsymbol{w}, X)$ does not change X in the cases we consider in this paper, Algorithm 1 can be extended to allow transformations on both parameters and data. The group actions on data during teleportations can be precomposed and applied to the input data at inference time.

The teleportation schedule K is a hyperparameter that determines when to perform teleportation. We define K as a set to allow flexible teleportation schedules, such as with non-fixed frequencies or teleporting only at the earlier epochs.

4.4 Symmetry Groups of Certain Optimization Problems

We give a few practical examples to demonstrate how teleportation can be used to accelerate optimization. Specifically, we first consider two test functions which are often used to evaluate optimization algorithms [15]. We then derive the symmetries of multi-layer neural networks.

4.4.1 Test Functions

Rosenbrock function.

The Rosenbrock function originally introduced by [120] has a characteristic global minimum that is inside a long, narrow, parabolicly-shaped flat valley. Finding the valley is easy but reaching the minimum is difficult. On a 2-dimensional space, the Rosenbrock function has the following form:

$$L_r(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2. \tag{4.3}$$

Booth function.

The Booth function [71] is also defined on $\mathbb{R}^2$ and has one global minimum at $(1,3)$ where the function evaluates to 0:

$$L_b(x_1,x_2) = (x_1+2x_2-7)^2+(2x_1+x_2-5)^2. \tag{4.4}$$

The following proposition identifies the symmetry of these two test functions.

Proposition 4.4.1. *The Rosenbrock and Booth functions have rotational symmetry. In other words, there exist action maps $a_r, a_b : \mathrm{SO}(2)\times\mathbb{R}^2 \to \mathbb{R}^2$, such that for all $g \in \mathrm{SO}(2)$,*

$$L_r(x_1,x_2) = L_r(a_r(g,[x_1,x_2])) \quad \textit{and} \quad L_b(x_1,x_2) = L_b(a_b(g,[x_1,x_2])).$$

Proof. Consider the Rosenbrock function with 2 variables [120]:

$$L(x_1,x_2) = 100(x_1^2-x_2)^2+(x_1-1)^2 \tag{4.5}$$

Let $u = 10(x_1^2-x_2)$ and $v = x_1-1$. After changing the variables from x and y to u and v, L has a rotational symmetry. Note that the function, $h : \mathbb{R}^2 \to \mathbb{R}^2$, that maps x_1, x_2 to u, v is bijective:

$$\begin{aligned}(u,v) = h(x_1,x_2) &= (10(x_1^2-x_2), x_1-1)\\ (x_1,x_2) = h^{-1}(u,v) &= (v+1,(v+1)^2-0.1u)\\ h(x_1,x_2) = h(y_1,y_2) &\Rightarrow (x_1,x_2) = (y_1,y_2)\end{aligned} \tag{4.6}$$

Next, we show that $SO(2)$ is a symmetry of $L(x_1,x_2)$. Let ρ be a representation of $SO(2)$ acting on $\mathbb{R}^2$. For $g \in SO(2)$, define the following group action:

$$g\cdot(x_1,x_2) = h^{-1}(\rho(g)h(x_1,x_2)) \tag{4.7}$$

Then

$$L(x_1, x_2) = L(g \cdot (x_1, x_2)) \tag{4.8}$$

For the Rosenbrock function with 2N parameters, we can construct a bijective function $h : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ by transforming each of the N pairs of variables as before, and $SO(2N)$ is a symmetry of $L(x_1, ..., x_{2N})$. However, we will only use the 2 variable version in the experiments.

Consider the Booth function [71]:

$$L(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Similar to the Rosebrock function, a change of variables reveals a rotational symmetry of L:

$$\begin{aligned}(u, v) &= h(x_1, x_2) = (x_1 + 2x_2 - 7, 2x_1 + x_2 - 5) \\ (x_1, x_2) &= h^{-1}(u, v) = (-\frac{1}{3}u + \frac{2}{3}v + 1, \frac{2}{3}u - \frac{1}{3}v + 3)\end{aligned} \tag{4.9}$$

The function $h : \mathbb{R}^2 \to \mathbb{R}^2$ that maps x_1, x_2 to u, v is bijective. Let ρ be a representation of $SO(2)$ acting on $\mathbb{R}^2$. For $g \in SO(2)$, define the following group action:

$$g \cdot (x_1, x_2) = h^{-1}(\rho(g) h(x_1, x_2)) \tag{4.10}$$

Then $L(x_1, x_2)$ admits an $SO(2)$ symmetry:

$$L(x_1, x_2) = L(g \cdot (x_1, x_2)) \tag{4.11}$$

□

During the teleportation step, our goal is to maximize the gradient within a level set of

the loss: $\max_{g\in \mathrm{SO}(2)} \|\frac{dL(g\cdot(x_1,x_2))}{dt}\|$.

4.4.2 Multi-Layer Neural Networks

Next, we consider feed-forward neural networks. Denote the output of the mth layer by $h_m \in \mathbb{R}^{d_m\times n}$, where d_m is the hidden dimension and n is the number of samples. Denote the input by $h_0 = X \in \mathbb{R}^{d_0\times n}$. In terms of the previous layer output $\tilde{h}_m = W_m h_{m-1}$ where $W_m \in \mathbb{R}^{d_m\times d_{m-1}}$ (we absorb biases into W_m by adding an extra row of ones in $\tilde{h}_{m-1}$), the output h_m is

$$h_m = \sigma(\tilde{h}_m) = \sigma\left(W_m h_{m-1}\right). \tag{4.12}$$

We assume the activation functions $\sigma : \mathbb{R} \to \mathbb{R}$ are bijections. For instance, Leaky-ReLU is bijective. For other activation, such Sigmoid or Tanh, we analytically extend them to bijective functions (e.g. $\tanh(x) + e^{x-N} - e^{-x+N}$ for $N \gg 1$). To define a symmetry in this case, we want to find $g \cdot W_m$ that keep the outputs h_k for $k \neq m-1$ invariant.

Proposition 4.4.2. *Assume that h_{m-2} is invertible. A multi-layer network with bijective activation σ has a $\mathrm{GL}_{d_{m-1}}$ symmetry. For $g_m \in G_m = \mathrm{GL}_{d_{m-1}}(\mathbb{R})$ the following group action keeps h_p with $p \geq m$ invariant*

$$g_m \cdot W_k = \begin{cases} W_m g_m^{-1} & k = m \\ \sigma^{-1}\left(g_m \sigma\left(W_{m-1} h_{m-2}\right)\right) h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases} \tag{4.13}$$

Proof. Under the given assumptoms, the map equation 4.13 satisfies all group action axioms. That is,

$$I \cdot W_k = \begin{cases} W_m I & k = m \\ \sigma^{-1}\left(I \sigma\left(W_{m-1} h_{m-2}\right)\right) h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases}$$

$$= W_k \tag{4.14}$$

and

$$\begin{aligned} g_1 \cdot (g_2 \cdot W_k) &= \begin{cases} W_m g_2^{-1} g_1^{-1} & k = m \\ \sigma^{-1}\left(g_1 \sigma\left(\left[\sigma^{-1}\left(g_2 \sigma\left(W_{m-1} h_{m-2}\right)\right) h_{m-2}^{-1}\right] h_{m-2}\right)\right) h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases} \\ &= \begin{cases} W_m (g_1 g_2)^{-1} & k = m \\ \sigma^{-1}\left((g_1 g_2) \sigma\left(W_{m-1} h_{m-2}\right)\right) h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases} \\ &= (g_1 g_2) \cdot W_k \end{aligned} \tag{4.15}$$

□

Note that this group action depends on the input to the network as well as the current weights of all the lower layers. Yet, since the action keeps the output of upper and lower layers invariant, multiple G_m for different m applied at the same time still keep the network output invariant. The proposition assumes that h_{m-2} is square and full-rank. When $n < d_{m-2}$ and h_{m-2} has rank n, equation 4.13 (with left inverse of h_{m-2}) keeps the loss invariant but does not satisfy the identity axiom of a group action.

Below we list an alternative group action on this architecture. If $\sigma\left(g_m W_{m-1} h_{m-2}\right)$ is invertible, for $g_m \in \mathrm{GL}_{d_{m-1}}(\mathbb{R})$, the following transformation is a loss-preserving group action:

$$g_m \cdot W_k = \begin{cases} W_m \sigma\left(W_{m-1} h_{m-2}\right) \sigma\left(g_m W_{m-1} h_{m-2}\right)^{-1} & k = m \\ g_m W_{m-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases} \tag{4.16}$$

Usually, the assumption is not guaranteed to hold for all group elements and all weights and data. Hence the above transformation may not preserve loss or be a valid group action. Nevertheless, we observe in practice that the change in the loss value is often small after such transformations on parameters. We therefore refer to equation (4.16) as an approximate symmetry and adopt it in the teleportation algorithm. Due to the possibility that $\sigma(g_m W_{m-1} h_{m-2})$ is not invertible, we use pseudoinverses in implementations.

4.5 Theoretical Analysis

In this section, we provide a theoretical analysis of teleportation. We show that with teleportation, SGD converges to a basin of stationary points. Building on its relation to Newton's method, teleportation leads to a mixture of linear and quadratic convergence. Lastly, in certain loss functions, one teleportation guarantees optimality of the entire gradient flow trajectory.

4.5.1 What Symmetries Help Accelerate Optimization

We now discuss the conditions that need to be satisfied for teleportation to accelerate GD. For brevity, we denote all trainable parameters (e.g. $\{W_1, \cdots W_p\}$ for the p-layer neural network) by a single flattened vector $\boldsymbol{w} \in \mathbb{R}^n$. Consider a symmetry G of the loss function $L(\boldsymbol{w})$, meaning for all $g \in G$, $L(\boldsymbol{w}) = L(g \cdot \boldsymbol{w})$. We quantify how teleportation by G changes the rate of loss decay, given by

$$\frac{dL(\boldsymbol{w})}{dt} = \left\langle \frac{\partial L}{\partial \boldsymbol{w}}, \frac{d\boldsymbol{w}}{dt} \right\rangle = -[\nabla L]^T \eta \nabla L = -\|\nabla L\|_\eta^2, \tag{4.17}$$

where η is the matrix of learning rates (which must be positive semi-definite) and $\|v\|_\eta^2 = v^T \eta v$ is the Mahalanobis norm. A constant learning rate means $\eta = I$.

The following proposition provides the condition a symmetry needs to satisfy to accelerate optimization.

Proposition 4.5.1. *Let $\boldsymbol{w}' = g \cdot \boldsymbol{w}$ be a point we teleport to. Let $J = \partial \boldsymbol{w}'/\partial \boldsymbol{w}$ be the Jacobian. Symmetry teleportation using g accelerates the rate of decay in L if it satisfies*

$$\left\| \left[J^{-1}\right]^T \nabla L(\boldsymbol{w}) \right\|_\eta^2 > \|\nabla L(\boldsymbol{w})\|_\eta^2 . \tag{4.18}$$

Proof. Let $\boldsymbol{w}' = g \cdot \boldsymbol{w}$. Denote the Jacobian as J, where $J_{ij} = \partial w_i'/\partial w_j$. Then the inverse of J has entries $J_{ij}^{-1} = \partial w_i/\partial w_j'$.

The gradient at $\boldsymbol{w}'$ is

$$\begin{aligned} \frac{\partial L(\boldsymbol{w}')}{\partial \boldsymbol{w}'} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}'} = \sum_j \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}_j} \frac{\partial \boldsymbol{w}_j}{\partial \boldsymbol{w}_i'} = \sum_j \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}_j} J_{ji}^{-1} = \left(\left(\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \right)^T J^{-1} \right)^T \\ = (J^{-1})^T \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \end{aligned} \tag{4.19}$$

The rate of change of L in gradient flow is

$$\frac{dL(\boldsymbol{w}')}{dt} = \left\langle \frac{\partial L}{\partial \boldsymbol{w}'}, \frac{d\boldsymbol{w}'}{dt} \right\rangle = - \left\| (J^{-1})^T \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \right\|_\eta^2 \tag{4.20}$$

Thus we will have a speedup if

$$\left\| (J^{-1})^T \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \right\|_\eta^2 > \left\| \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \right\|_\eta^2 \tag{4.21}$$

□

If the action of the symmetry group $G \subset GL_n$ is linear we have $\boldsymbol{w}' = g \cdot \boldsymbol{w} = g\boldsymbol{w}$ and $J = g$. It follows that if G is a subgroup of the orthogonal group, dL/dt will be invariant:

Corollary 4.5.2. *Let O_η denote the orthogonal group of invariances of the inverse of the learning rate, η^{-1}, meaning for $g \in O_\eta$, $g^T \eta^{-1} g = \eta^{-1}$. Then*

$$\forall g \in O_\eta, \qquad \frac{dL(g \cdot \boldsymbol{w})}{dt} = \frac{L(\boldsymbol{w})}{dt}. \tag{4.22}$$

Proof. Since $J = g$, using equation 4.17 and the l.h.s. of equation 4.18 we have

$$\frac{dL(g \cdot \boldsymbol{w})}{dt} = -\nabla L^T g^{-1} \eta \left[g^{-1}\right]^T \nabla L = \nabla L^T \left[g^T \eta^{-1} g\right]^{-1} \nabla L = \|\nabla L\|_\eta^2 \tag{4.23}$$

□

In the simple case where the learning rate is a constant, $O_\eta = O(n)$ becomes the classic orthogonal group (e.g. rotations) with $g^T g = I$. In general, when g preserves the norm of the gradient, symmetry teleportation has no effect on the convergence rate.

From Theorem 3.2 in [61], assuming that L is β-smooth and is bounded by $|L| \leq M$, the gradient norm in gradient descent converges as $\frac{1}{2\beta} \sum_{t=1}^{T} \|(\nabla L)|_{\boldsymbol{w}_t}\| \leq 2M$. Teleportation increases $(\nabla L)|_{\boldsymbol{w}_t}$, therefore requiring less time to reach convergence.

4.5.2 Improvement of Subsequent Steps

Since teleportation moves the parameters to a point with a larger gradient, and subsequent GD steps are local, we would expect that teleportation improves the magnitude of the gradient for a few future steps as well. The following results formalize this intuition.

Assumption 4.5.3 (Lipschitz Continuity)**.** *The l2 norm of the gradient is Lipschitz continuous with constant $L \in \mathbb{R}^{\geq 0}$, which is*

$$\left| \left\| \frac{\partial L}{\partial \boldsymbol{w}_1} \right\|_2 - \left\| \frac{\partial L}{\partial \boldsymbol{w}_2} \right\|_2 \right| \leq L \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_2, \tag{4.24}$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2$ are two points in the parameter space and L is the Lipschitz constant.

Proposition 4.5.4. *Consider the gradient descent with a G-invariant loss $L(\boldsymbol{w})$ and learning rate $\eta \in \mathbb{R}^+$. Let $\boldsymbol{w}_t$ be the parameter at time t and $\boldsymbol{w}'_t = g \cdot \boldsymbol{w}_t$ the parameter teleported by $g \in G$. Let $\boldsymbol{w}_{t+T}$ and $\boldsymbol{w}'_{t+T}$ be the parameters after T more steps of gradient descent from $\boldsymbol{w}_t$ and $\boldsymbol{w}'_t$*

respectively. Under Assumption 4.5.3, if $\eta L < 1$*, and*

$$\frac{\|\partial L/\partial \boldsymbol{w}_t'\|_2}{\|\partial L/\partial \boldsymbol{w}_t\|_2} \geq \frac{(1+\eta L)^T}{(1-\eta L)^T}, \tag{4.25}$$

then

$$\left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+T}'} \right\|_2 \geq \left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+T}} \right\|_2. \tag{4.26}$$

Proof. From the definition of Lipschitz continuity and the update rule of gradient descent,

$$\left| \left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+1}} \right\|_2 - \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \right| \leq L\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_2 = L \left\| \eta \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \tag{4.27}$$

Equivalently,

$$(1-\eta L) \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \leq \left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+1}} \right\|_2 \leq (1+\eta L) \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \tag{4.28}$$

By unrolling T steps, we have

$$(1-\eta L)^T \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \leq \left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+T}} \right\|_2 \leq (1+\eta L)^T \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \tag{4.29}$$

Similarly, for a teleported point $\boldsymbol{w}_t' = g \cdot \boldsymbol{w}_t$,

$$(1-\eta L)^T \left\| \frac{\partial L}{\partial \boldsymbol{w}_t'} \right\|_2 \leq \left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+T}'} \right\|_2 \leq (1+\eta L)^T \left\| \frac{\partial L}{\partial \boldsymbol{w}_t'} \right\|_2 \tag{4.30}$$

Therefore, if

$$(1-\eta L)^T \left\| \frac{\partial L}{\partial \boldsymbol{w}_t'} \right\|_2 \geq (1+\eta L)^T \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2 \tag{4.31}$$

then it is guaranteed that

$$\left\| \frac{\partial L}{\partial \boldsymbol{w}'_{t+T}} \right\|_2 \geq \left\| \frac{\partial L}{\partial \boldsymbol{w}_{t+T}} \right\|_2 \tag{4.32}$$

□

Proposition 4.5.4 provides a sufficient condition for teleportation to improve T future steps. The condition is met when L is small, η is small, or the initial improvement, $\left\| \frac{\partial L}{\partial \boldsymbol{w}'_t} \right\|_2 / \left\| \frac{\partial L}{\partial \boldsymbol{w}_t} \right\|_2$, is large.

4.5.3 Convergence Analysis for Convex Quadratic Functions

Teleportation improves the magnitude of gradient for the current step. We have also shown that the magnitude of gradient stays large for a few subsequent steps (Proposition 4.5.4). In this section, we analyze a class of functions where teleporting once guarantees optimality at all future times. We consider a trajectory optimal if for every point on the trajectory, the magnitude of gradient is at a local maximum in the loss level set that contains the point.

Consider a positive definite quadratic form $L_A(\boldsymbol{w}) = \boldsymbol{w}^T A \boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{R}^n$ is the parameter and $A \in \mathbb{R}^{n \times n}$ is a positive definite matrix. The gradient of L_A is $\nabla L_A = 2A\boldsymbol{w}$, and the Hessian of L_A is $H = 2A$. Since A is defined to be positive definite, L_A is convex. The function L_A has global minimum at a single point $\boldsymbol{w}^* = 0$.

Let ρ be a representation of $O(n)$ acting on $\mathbb{R}^n$. For $g \in O(n)$, we define the following group action:

$$g \cdot \boldsymbol{w} = A^{-\frac{1}{2}} \rho(g) A^{\frac{1}{2}} \boldsymbol{w}. \tag{4.33}$$

Then it can be shown that $L_A(\boldsymbol{w})$ admits a $O(n)$ symmetry:

$$L_A(g \cdot \boldsymbol{w}) = \boldsymbol{w}^T {A^{\frac{1}{2}}}^T \rho(g)^T {A^{-\frac{1}{2}}}^T A A^{-\frac{1}{2}} \rho(g) A^{\frac{1}{2}} \boldsymbol{w} = \boldsymbol{w}^T A \boldsymbol{w} = L_A(\boldsymbol{w}). \tag{4.34}$$

Let $S_c = \{\boldsymbol{w} : L_A(\boldsymbol{w}) = c\}$ be a level set of L_A. We will show that after a teleportation, every point on the gradient flow trajectory is optimal in its level set.

We first show that starting from a point in S_c, all other points in S_c can be reached with one teleportation.

Lemma 4.5.5. *S_c contains a single orbit. That is, $G \cdot \boldsymbol{w} \equiv \{g \cdot \boldsymbol{w} : g \in G\} = S_c$ for all $\boldsymbol{w} \in S_c$.*

Proof. Consider two points $\boldsymbol{w}_1, \boldsymbol{w}_2 \in S_c$. Then $\boldsymbol{w}_1^T A \boldsymbol{w}_1 = (A^{\frac{1}{2}}\boldsymbol{w}_1)^T (A^{\frac{1}{2}}\boldsymbol{w}_1) = c$ and $\boldsymbol{w}_2^T A \boldsymbol{w}_2 = (A^{\frac{1}{2}}\boldsymbol{w}_2)^T (A^{\frac{1}{2}}\boldsymbol{w}_2) = c$. Let $\boldsymbol{v}_1 = \frac{A^{\frac{1}{2}}\boldsymbol{w}_1}{\sqrt{c}}$, $\boldsymbol{v}_2 = \frac{A^{\frac{1}{2}}\boldsymbol{w}_2}{\sqrt{c}}$ and $\mathbf{e}_1 = [1, 0, ..., 0]^T$. Since $\|v_1\| = \|v_2\| = 1$, there exists $g_1, g_2 \in O(n)$, such that $g_1 \mathbf{e}_1 = \boldsymbol{v}_1$ and $g_2 \mathbf{e}_1 = \boldsymbol{v}_2$. One way to construct such g_1 is let the first column be equal to $\boldsymbol{v}_1$ and other columns be the rest of the orthonormal basis. Let $g = g_2 g_1^{-1}$. Then $\boldsymbol{v}_2 = g\boldsymbol{v}_1$, $A^{\frac{1}{2}}\boldsymbol{w}_2 = gA^{\frac{1}{2}}\boldsymbol{w}_1$, and $\boldsymbol{w}_2 = A^{-\frac{1}{2}} g A^{\frac{1}{2}} \boldsymbol{w}_1 = g \cdot \boldsymbol{w}_1$.

We have shown that for any $\boldsymbol{w}_1, \boldsymbol{w}_2 \in S_c$, there exists a $g \in G$ such that $\boldsymbol{w}_2 = g \cdot \boldsymbol{w}_1$. Therefore, the group action of G on S_c is transitive. Equivalently, S_c contains a single orbit. □

The objective of teleportation is transforming parameters using a group element to maximize the norm of gradient:

$$\max_{g \in G} \|\nabla L_A|_{g \cdot \boldsymbol{w}}\|_2^2. \tag{4.35}$$

Since all points on the level set are reachable, the target teleportation destination is the point with the largest gradient norm on the same level set. In other words, equation 4.35 is equivalent to the following optimization problem:

$$\begin{aligned} &\max_{\boldsymbol{w}'} \|\nabla L_A|_{\boldsymbol{w}'}\|_2^2 \\ &\text{s.t. } L_A(\boldsymbol{w}') = L_A(\boldsymbol{w}). \end{aligned} \tag{4.36}$$

Let $c = L_A(\boldsymbol{w})$. Substitute in L_A and ∇L_A, we have the following equivalent formulation:

$$\begin{aligned} &\max_{\boldsymbol{w}'} \|A\boldsymbol{w}'\|_2^2 \\ &\text{s.t. } \boldsymbol{w}'^T A\boldsymbol{w}' = c. \end{aligned} \tag{4.37}$$

Next, we solve this optimization problem and show that the gradient norm is maximized on the gradient flow trajectory starting from its solution.

Lemma 4.5.6. *The solution to equation 4.37 is an eigenvector of A corresponding to its largest eigenvalue.*

Proof. We solve equation 4.37 using the method of Lagrangian multipliers. The Lagrangian of equation 4.37 is

$$\mathscr{L} = \boldsymbol{w}'^T A^T A\boldsymbol{w}' - \lambda(\boldsymbol{w}'^T A\boldsymbol{w}' - c). \tag{4.38}$$

Setting the derivative with respect of $\boldsymbol{w}'$ to 0, we have

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{w}} = 2A^T A\boldsymbol{w}' - 2\lambda A\boldsymbol{w}' = 0, \tag{4.39}$$

which gives

$$A^T A\boldsymbol{w}' = \lambda A\boldsymbol{w}'. \tag{4.40}$$

Since A is positive definite, $A = A^T$ and A is invertible. Therefore,

$$A\boldsymbol{w}' = \lambda \boldsymbol{w}', \tag{4.41}$$

so the solution to equation 4.37 is an eigenvector of A. Then, the constraint is $\boldsymbol{w}'^T A\boldsymbol{w}' = \lambda\|\boldsymbol{w}'\|^2 = c$, and the objective becomes $\max_{\boldsymbol{w}'} \lambda^2\|\boldsymbol{w}'\|^2 = \max_{\boldsymbol{w}'} c\lambda$. Therefore, we want λ to

be the largest eigenvalue of A. Hence the optimal $\boldsymbol{w}'$ is an eigenvector of A corresponding to its largest eigenvalue. □

After a teleportation, every point on the gradient flow trajectory is optimal in its level set.

Proposition 4.5.7. *If at point $\boldsymbol{w}$, $\|\nabla L_A|_{\boldsymbol{w}}\|^2$ is at a maximum in $S_{L_A(\boldsymbol{w})}$, then for any point $\boldsymbol{w}'$ on the gradient flow trajectory starting from $\boldsymbol{w}$, $\|\nabla L_A|_{\boldsymbol{w}'}\|^2$ is at a maximum in $S_{L_A(\boldsymbol{w}')}$.*

Proof. From Proposition 4.5.6, $\boldsymbol{w}$ is an eigenvector of A corresponding to its largest eigenvalue. Then the gradient of L_A is $A\boldsymbol{w} = \lambda \boldsymbol{w}$. Therefore, on the gradient flow trajectory starting from $\boldsymbol{w}$, every point has the same direction as $\boldsymbol{w}$, meaning that the points are all eigenvectors of A corresponding to its largest eigenvalue. Therefore, $\|\nabla L_A\|^2$ is always at a maximum on the loss level sets. □

Finally, we observe that for L_A, teleportation moves the parameters closer to the global minimum in Euclidean distance. In other words, maximizing the magnitude of gradient minimizes the distance to $\boldsymbol{w}^*$ in a loss level set.

Proposition 4.5.8. *Consider a point $\boldsymbol{w}$ in the parameter space. Let $g = \arg\max_{g \in G} \|\nabla L_A|_{g \cdot \boldsymbol{w}}\|_2^2$. Then $g \cdot \boldsymbol{w} = \arg\min_{\boldsymbol{w}' \in S_{L_A(\boldsymbol{w})}} \|\boldsymbol{w}' - \boldsymbol{w}^*\|_2^2$.*

Proof. Similar to Proposition 4.5.6, we solve this optimization problem using the method of Lagrangian multipliers. Note that $\boldsymbol{w}^* = 0$. The Lagrangian is

$$\mathscr{L} = \boldsymbol{w}'^T \boldsymbol{w}' - \lambda(\boldsymbol{w}'^T A \boldsymbol{w}' - c). \tag{4.42}$$

Setting the derivative with respect of $\boldsymbol{w}'$ to 0, we have

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{w}} = 2\boldsymbol{w}' - 2\lambda A \boldsymbol{w}' = 0, \tag{4.43}$$

which gives

$$A\boldsymbol{w}' = \lambda \boldsymbol{w}', \tag{4.44}$$

so the solution to equation 4.37 is an eigenvector of A. Then, the constraint is $\boldsymbol{w}'^T A\boldsymbol{w}' = \lambda \|\boldsymbol{w}'\|^2 = c$, and the objective becomes $\min_{\boldsymbol{w}'} \|\boldsymbol{w}'\|^2 = \min_{\boldsymbol{w}'} \frac{c}{\lambda}$. Therefore, we want λ to be the largest eigenvalue of A. Hence the optimal $\boldsymbol{w}'$ is an eigenvector of A corresponding to its largest eigenvalue, which is the same as the solution to equation 4.37. □

For a more concrete example, consider a diagonal matrix A with positive diagonal elements. Then the level sets of L_A are n-dimensional ellipsoids centered at the origin 0, with axes in the same directions as the standard basis. The point with largest $\|\nabla L_A\|^2$ on a level set is in the eigendirection of A corresponding to its largest eigenvalue, or equivalently, a point on the smallest semi-axes of the ellipsoid. Note that this point has the smallest distance to the global minimum $\boldsymbol{w}^* = 0$ among all points in the same level set. In addition, the gradient flow trajectory from this point always points to $\boldsymbol{w}^*$. Therefore, like the 2D ellipse function, one teleportation on the n-dimensional ellipsoid also guarantees optimal gradient norm at all points on the trajectory.

4.5.4 Improved Convergence Bound for SGD

At each iteration $t \in \mathbb{N}^+$ in SGD, we choose a group element $g^t \in G$ and use teleportation before each gradient step as follows

$$\boldsymbol{w}^{t+1} = g^t \cdot \boldsymbol{w}^t - \eta \nabla L(g^t \cdot \boldsymbol{w}^t, \xi^t). \tag{4.45}$$

Here η is a learning rate, $\nabla L(\boldsymbol{w}^t, \xi^t)$ is the gradient of $L(\boldsymbol{w}^t, \xi^t)$ with respect to the parameters $\boldsymbol{w}$, and $\xi^t \sim \mathscr{D}$ is a mini-batch of data sampled i.i.d from the data distribution at each iteration.

By choosing the group element that maximizes the gradient norm, we show in the following theorem that the iterates in equation 4.45 converge to a basin of stationary points,

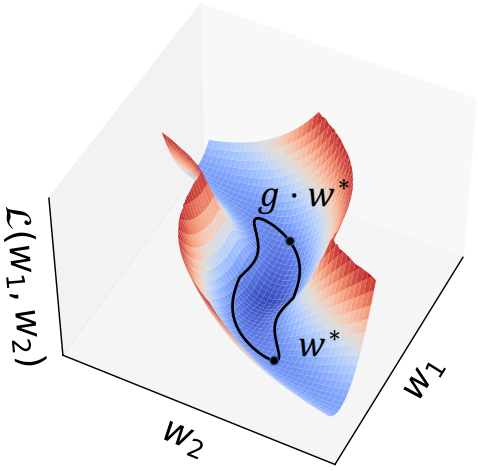


Figure 4.2. With teleportation, SGD converges to a basin where all points on the level set are stationary points.

where all points that can be reached via teleportation are also stationary points (visualized in Figure 4.2).

Theorem 4.5.9. *(Smooth non-convex) Let $L(\boldsymbol{w}, \xi)$ be β–smooth and let*

$$\sigma^2 \stackrel{def}{=} L(\boldsymbol{w}^*) - \mathbb{E}\left[\inf_{\boldsymbol{w}} L(\boldsymbol{w}, \xi)\right].$$

Consider the iterates $\boldsymbol{w}^t$ given by equation 4.45 where

$$g^t \in \arg\max_{g \in G} \|\nabla L(g \cdot \boldsymbol{w}^t)\|^2,$$

which we assume exists. [1] *If $\eta = \frac{1}{\beta\sqrt{T-1}}$ then*

$$\begin{aligned} &\min_{t=0,\ldots,T-1} \mathbb{E}\left[\max_{g \in G} \|\nabla L(g \cdot \boldsymbol{w}^t)\|^2\right] \\ &\qquad \leq \frac{2\beta}{\sqrt{T-1}} \mathbb{E}\left[L(\boldsymbol{w}^0) - L(\boldsymbol{w}^*)\right] + \frac{\beta\sigma^2}{\sqrt{T-1}}, \end{aligned} \tag{4.46}$$

where the expectation is the total expectation with respect to the data ξ^t for $t = 0, \ldots, T-1$.

[1] For instance when G is compact and $\|\nabla L(g \cdot \boldsymbol{w}^t)\|$ is continuous over G, or when the gradient is a coercive

This theorem is an improvement over vanilla SGD, for which we would have instead that

$$\min_{t=0,\ldots,T-1} \mathbb{E}\left[\|\nabla L(\boldsymbol{w}^t)\|^2\right] \leq \frac{2\beta}{\sqrt{T-1}} \mathbb{E}\left[L(\boldsymbol{w}^0) - L(\boldsymbol{w}^*)\right] + \frac{\beta\sigma^2}{\sqrt{T-1}}.$$

The above only guarantees that there exists a single point $\boldsymbol{w}^t$ for which the gradient norm will eventually be small. In contrast, our result in equation 4.46 guarantees that for all points over the orbit $\{g \cdot \boldsymbol{w}^t \ : \ \forall g \in G\}$, the gradient norm will be small. For strictly convex loss functions, $\max_{g\in G}\|\nabla L(g \cdot \boldsymbol{w})\|^2$ is non-decreasing with $L(\boldsymbol{w})$. In this case, the value of L is smaller after T steps of SGD with teleportation, compared to vanilla SGD (Proposition C.2.2).

4.5.5 Relation to Second-Order Optimization

Since our algorithm involves optimizing the gradients, symmetry teleportation is closely related to second-order optimization methods. In this section, we show that the target point to teleport to, gradient descent becomes equivalent to Newton's method.

We first note that when the norm of the gradient is at a critical point on the level set of the loss function, the gradient is an eigenvector of the Hessian.

Lemma 4.5.10. *If* $\partial_{\boldsymbol{v}} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = 0$ *for all unit vector* $\boldsymbol{v}$ *that is orthogonal to* $\frac{\partial L}{\partial \boldsymbol{w}}$*, then* $\frac{\partial L}{\partial \boldsymbol{w}}$ *is an eigenvector of the Hessian of L.*

Proof. From the definition of the directional derivative,

$$\partial_{\boldsymbol{v}} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = \boldsymbol{v} \cdot \frac{\partial}{\partial \boldsymbol{w}} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 \tag{4.47}$$

Writing the last term in indices,

$$\frac{\partial}{\partial \boldsymbol{w}_i} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = \frac{\partial}{\partial \boldsymbol{w}_i} \sum_j \left(\frac{\partial L}{\partial \boldsymbol{w}_j} \right)^2$$

function and G is bounded.

$$
\begin{aligned}
&= \sum_j \frac{\partial}{\partial \boldsymbol{w}_i} \left(\frac{\partial L}{\partial \boldsymbol{w}_j} \right)^2 \\
&= \sum_j 2 \frac{\partial L}{\partial \boldsymbol{w}_j} \frac{\partial^2 L}{\partial \boldsymbol{w}_i \partial \boldsymbol{w}_j} \\
&= 2 \left(H \frac{\partial L}{\partial \boldsymbol{w}} \right)_i
\end{aligned} \tag{4.48}
$$

Removing the indices,

$$
\frac{\partial}{\partial \boldsymbol{w}} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = 2H \frac{\partial L}{\partial \boldsymbol{w}} \tag{4.49}
$$

Substitute back and we have

$$
\partial_{\boldsymbol{v}} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = \boldsymbol{v} \cdot \left(2H \frac{\partial L}{\partial \boldsymbol{w}} \right) \tag{4.50}
$$

Since $\partial_{\boldsymbol{v}} \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = 0$ for all vector $\boldsymbol{v}$ that is orthogonal to $\frac{\partial L}{\partial \boldsymbol{w}}$, $\boldsymbol{v} \cdot \left(2H \frac{\partial L}{\partial \boldsymbol{w}} \right) = 0$ for all vector $\boldsymbol{v}$ that is orthogonal to $\frac{\partial L}{\partial \boldsymbol{w}}$. In other words, $2H \frac{\partial L}{\partial \boldsymbol{w}}$ is orthogonal to all vectors that are orthogonal to $\frac{\partial L}{\partial \boldsymbol{w}}$. Therefore, $2H \frac{\partial L}{\partial \boldsymbol{w}}$ has the same direction of $\frac{\partial L}{\partial \boldsymbol{w}}$, and $\frac{\partial L}{\partial \boldsymbol{w}}$ is an eigenvector of the Hessian of L. □

As a direct consequence of Lemma 4.5.10, the gradient direction and Newton's direction aligns at the the teleportation target point.

Proposition 4.5.11. *Let $S_{L_0} = \{\boldsymbol{w} : L(\boldsymbol{w}) = L_0\}$ be a level set of L. If at a particular $\boldsymbol{w} \in S_{L_0}$ we have $\|\nabla L(\boldsymbol{w})\|_2 \geq \|\nabla L(\boldsymbol{w}')\|_2$ for all $\boldsymbol{w}'$ in a small neighborhood of $\boldsymbol{w}$ in S_{L_0}, then the gradient $\nabla L(\boldsymbol{w})$ has the same direction as the direction from Newton's method, $H^{-1}\nabla L(\boldsymbol{w})$.*

Proof. From Lemma 4.5.10, $\frac{dL}{dw}$ is an eigenvector of H. Therefore, it is also an eigenvector of H^{-1}. Hence $\frac{dL}{dw}$ has the same direction as $H^{-1} \frac{dL}{dw}$. □

Proposition 4.5.11 provides an alternative way to interpret teleportation. Instead of computing the Newton's direction, we search within the loss level set for a point where the

gradient has the same direction as Newton's direction. However, symmetry teleportation does not require computing the full Hessian matrix. The second derivative required for optimizing over continuous groups is obtained by taking derivatives with respect to parameters and then with respect to the group element, as opposed to taking the derivative with respect to parameters twice. This makes the computation significantly more feasible than Newton's method on neural networks with large number of parameters.

We can leverage the convergence of Newton's method to derive the convergence rate of teleportation for the deterministic setting.

Proposition 4.5.12 (Quadratic term in convergence rate)**.** *Let L be strictly convex and let $\boldsymbol{w}_0 \in \mathbb{R}^d$. Let*

$$\boldsymbol{w}' \in \arg\max_{\boldsymbol{w}\in\mathbb{R}^d} \frac{1}{2}\|\nabla L(\boldsymbol{w})\|^2, \quad \text{s.t.} \quad L(\boldsymbol{w}) = L(\boldsymbol{w}_0).$$

Let $\nabla^2 L$ be the Hessian of L, and $\lambda_{\max}(\nabla^2 L(\boldsymbol{w}))$ be the largest eigenvalue of $\nabla^2 L(\boldsymbol{w})$. If $\nabla L(\boldsymbol{w}') \neq 0$, then there exists λ_0 such that $0 \leq \lambda_0 \leq \lambda_{\max}(\nabla^2 L(\boldsymbol{w}_0))$, and one step of gradient descent after teleportation with learning rate $\gamma > 0$ gives

$$\boldsymbol{w}_1 = \boldsymbol{w}' - \gamma \nabla L(\boldsymbol{w}') = \boldsymbol{w}' - \gamma\lambda_0 \nabla^2 L(\boldsymbol{w}')^{-1} \nabla L(\boldsymbol{w}'). \tag{4.51}$$

Let $\boldsymbol{w}' = g_0 \cdot \boldsymbol{w}_0$. If $\gamma \leq \frac{1}{\lambda_0}$, L is a μ–strongly convex L–smooth function, and the Hessian is G–Lipschitz, then we have that

$$\|\boldsymbol{w}_1 - \boldsymbol{w}^*\| \leq \frac{G}{2\mu}\|g_0 \cdot \boldsymbol{w}_0 - \boldsymbol{w}^*\|^2 + |1 - \gamma\lambda_0| \frac{L}{2\mu}\|g_0 \cdot \boldsymbol{w}_0 - \boldsymbol{w}^*\|.$$

More details about the assumptions and the proof are in Appendix C.3. Note that due to unknown step size λ_0, extra care is needed in establishing this convergence rate.

The above proposition shows that taking one step of teleportation and one gradient step,

the result is equal to taking a dampened Newton step (equation 4.51). Hence, the convergence rate has a quadratically contracting term $\|g_0 \cdot \boldsymbol{w}_0 - \boldsymbol{w}^*\|^2$, which is typical of second order methods. In particular, setting $\gamma = 1/\lambda_0$ we would have local quadratic convergence. In contrast, without the teleportation step and under the same assumptions, we would have the following linear convergence

$$\|\boldsymbol{w}_1 - \boldsymbol{w}^*\| \leq (1 - \mu\gamma) \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|$$

for $\gamma \leq \frac{1}{L}$ using gradient descent. Thus there would be no quadratically contracting term.

In practice, however, the group action used for teleportation is usually not transitive. Additionally, we do not teleport using the optimal group element since it can be unbounded. We also do not apply teleportation in every gradient descent step. Therefore, Proposition 4.5.11 serves as an intuition instead of an exact formulation of how teleportation works. We provide empirical evidence in Section 6 that these approximations do not erase the benefits of teleportation completely, and leave theoretical investigations of the connection to second-order methods under approximations as future work.

4.5.6 When is One Teleportation Enough

Despite the guaranteed improvement in convergence, teleporting before every gradient descent step is computationally expensive. Hence we teleport only occasionally. In fact, for certain optimization objectives, every point on the gradient flow has the largest gradient norm in its loss level set after one teleportation [157]. In past work, this result is limited to convex quadratic functions. In this section, we give a sufficient condition for when one teleportation results in an optimal trajectory for general loss functions. Full proofs can be found in Appendix C.4.

Let $V : \mathscr{M} \to T\mathscr{M}$ be a vector field on the manifold $\mathscr{M}$, where $T\mathscr{M}$ denotes the associated tangent bundle. Here we consider the parameter space $\mathscr{M} = \mathbb{R}^n$, although results in this section can be extended to optimization on other manifolds. In this case, we may write

$V = v^i \frac{\partial}{\partial w^i}$ using the component functions $v^i : \mathbb{R}^n \to \mathbb{R}$ and coordinates w^i.

Consider a smooth loss function $L : \mathscr{M} \to \mathbb{R}$. Let G be a symmetry group of L, i.e. $L(g \cdot \boldsymbol{w}) = L(\boldsymbol{w})$ for all $\boldsymbol{w} \in \mathscr{M}$ and $g \in G$. Let $\mathfrak{X}$ be the set of all vector fields on $\mathscr{M}$. Let $R = r^i \frac{\partial}{\partial w^i}$, where $r^i = -\frac{\partial L}{\partial w_i}$, be the reverse gradient vector field. Let $\mathfrak{X}_\perp = \{A = a^i \frac{\partial}{\partial w^i} \in \mathfrak{X} |\ a^i \in C^\infty(\mathscr{M})$ and $\sum_i a^i(\boldsymbol{w}) r^i(\boldsymbol{w}) = 0, \forall \boldsymbol{w} \in \mathscr{M}\}$ be the set of vector fields orthogonal to R. If G is a Lie group, the infinitesimal action of its Lie algebra $\mathfrak{g}$ defines a set of vector fields $\mathfrak{X}_\mathfrak{g} \subseteq \mathfrak{X}_\perp$.

A gradient flow is a curve $\gamma : \mathbb{R} \to \mathscr{M}$ where the velocity is given by the value of R, i.e. $\gamma'(t) = R_{\gamma(t)}$ for all $t \in \mathbb{R}$. The Lie bracket $[A,R]$ defines the derivative of R with respect to A. Flows of A and R commute if and only if $[A,R] = 0$ (Theorem 9.44, [90]). That is, teleportation can affect the convergence rate only if $[A,R]L \neq 0$ for some $A \in \mathfrak{X}_\mathfrak{g}$. To simplify notation, we write $([W,R]L)(\boldsymbol{w}) = 0$ for a set of vector fields $W \subseteq \mathfrak{X}$ when $([A,R]L)(\boldsymbol{w}) = 0$ for all $A \in W$.

We consider a gradient flow optimal if every point on the flow is a critical point of the magnitude of gradient in its loss level set. Note that this definition does not exclude the case where points on the flow are minimizers of the magnitude of gradient.

Definition 4.5.13. *Let $f : \mathscr{M} \to \mathbb{R}, \boldsymbol{w} \mapsto \left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2$. A point $\boldsymbol{w} \in M$ is optimal with respect to a set of vector fields $W \subseteq \mathfrak{X}_\perp$ if $Af(\boldsymbol{w}) = 0$ for all $A \in W$. A gradient flow $\gamma : \mathbb{R} \to \mathscr{M}$ is optimal with respect to W if $\gamma(t)$ is optimal with respect to W for all $t \in \mathbb{R}$.*

Proposition 4.5.14. *A point $\boldsymbol{w} \in \mathscr{M}$ is optimal with respect to a set of vector fields W if and only if $([W,R]L)(\boldsymbol{w}) = 0$.*

Proof. Note that $AL = a^i \frac{\partial L}{\partial w^i} = 0$. We have

$$[A,R]L = ARL - RAL = A\left(r^i \frac{\partial L}{\partial w_i}\right) - 0 = -A\left\| \frac{\partial L}{\partial \boldsymbol{w}} \right\|_2^2 = -Af. \tag{4.52}$$

The result then follows from Definition 4.5.13. □

A sufficient condition for one teleportation to result in an optimal trajectory is that whenever the function $[A,R]L$ vanishes at $\boldsymbol{w} \in \mathscr{M}$, it vanishes along the entire gradient flow

starting at $\boldsymbol{w}$.

Proposition 4.5.15. *Let $W \subseteq \mathfrak{X}_\perp$ be a set of vector fields that are orthogonal to $\frac{\partial L}{\partial \boldsymbol{w}}$. Assume that for all $\boldsymbol{w} \in \mathscr{M}$ such that $([W,R]L)(\boldsymbol{w}) = 0$, we have that $(R[W,R]L)(\boldsymbol{w}) = 0$. Then the gradient flow starting at any optimal point with respect to W is optimal with respect to W.*

Proof. Consider the gradient flow γ that starts at an optimal point in W. The derivative of $[A,R]L$ along γ is

$$\frac{d}{dt}[A,R]L(\gamma(t)) = \gamma'(t)([A,R]L)(\gamma(t)) = -R[A,R]L(\gamma(t)). \tag{4.53}$$

Since $\gamma(0)$ is an optimal point, $[A,R]L(\gamma(0)) = 0$ for all $A \in W$ by Proposition 4.5.14. By assumption, if $[A,R]L(\gamma(t)) = 0$ for all $A \in W$, then $R([A,R]L)(\gamma(t)) = 0$ for all $A \in W$. Therefore, both the value and the derivative of $[A,R]L$ stay 0 along γ. Since $[A,R]L(\gamma(t)) = 0$ for all $t \in \mathbb{R}$, γ is optimal in W. □

To help check when the assumption in Proposition 4.5.15 is satisfied, we provide an alternative form of $R[W,R]L(\boldsymbol{w})$ when $[W,R]L(\boldsymbol{w}) = 0$.

Proposition 4.5.16. *If at all optimal points in $S = \{(M\frac{\partial L}{\partial \boldsymbol{w}})^i \frac{\partial}{\partial w^i} \in \mathfrak{X} |\ M \in \mathbb{R}^{n\times n}, M^T = -M\}$,*

$$M^j_\alpha \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} = 0$$

for all anti-symmetric matrices $M \in \mathbb{R}^{n\times n}$, then the gradient flow starting at an optimal point in S is optimal in S.

From Proposition 4.5.16, we see that $R[W,R]L(\boldsymbol{w})$ is not automatically 0 when $[W,R]L(\boldsymbol{w})$ is 0. Therefore, even if the group is big enough to have its infinitesimal actions cover the tangent space of the level set ($\mathfrak{X}_\mathfrak{g} = \mathfrak{X}_\perp$), one teleportation does not guarantee that the gradient flow intersects all future level sets at optimal points. However, for loss functions that satisfy the condition in Proposition 4.5.15, teleporting once optimizes the entire trajectory. This is the

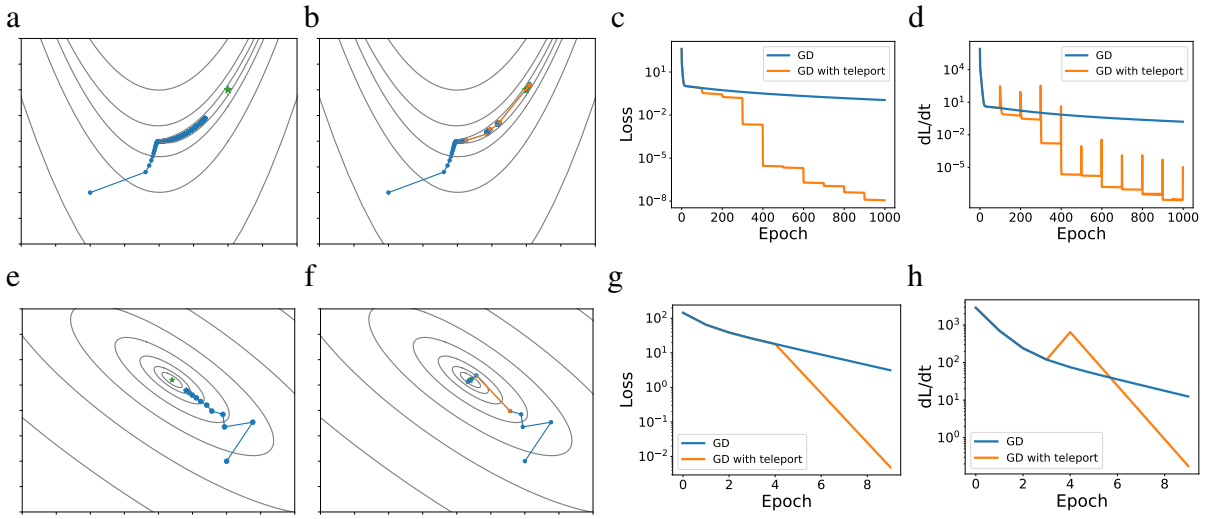


Figure 4.3. Optimization of the Rosenbrock function (top row) and Booth function (bottom row) using (a) gradient descent and (b) the proposed algorithm. Contours represent the level sets of the loss function. Loss L and convergence rate dL/dt are shown in (c) and (d). Teleportation helps move the parameters towards the target.

case, for example, when $\frac{\partial^3 L}{\partial w^k \partial w^i \partial w^j}\frac{\partial L}{\partial w^\alpha} = \frac{\partial^3 L}{\partial w^k \partial w^i \partial w^\alpha}\frac{\partial L}{\partial w^j}$ for all i,k,j,α (Proposition C.4.3). In particular, all quadratic functions meet this condition.

4.6 Experiments

4.6.1 Acceleration through Symmetry Teleportation

We examine the effect of symmetry teleportation on optimization. We illustrate teleportation in the parameter space on two test functions and show a speedup in regression and classification problems using multilayer neural networks. For test functions, we compared with GD for illustration purposes. For multi-layer neural networks, we also include AdaGrad as a more competitive baseline.

Rosenbrock function.

We apply symmetry teleportation to optimize the 2-variable Rosenbrock function equation 4.3. The parameters x_1, x_2 are initialized to $(-1,-1)$. Each algorithm is run 1000 steps with learning rate 10^{-3}. We teleport the parameters every 100 steps. The group elements are found

by gradient ascent on θ, the parameter for the $\mathrm{SO}(2)$ group, for 10 steps with learning rate 10^{-1}.

The trajectory of parameters and the loss level sets are plotted in Figure 4.3a,b. The blue star denotes the final position of parameters, the green star denotes the target, and orange dots are the positions from which symmetry transforms start. While gradient descent is not able to reach the target in 1000 steps, teleportation allows large steps and reaches the target much earlier. Figure 4.3d shows that teleportation improves the norm of gradients in the following step, and 4.3c shows its effect on the loss value. Teleportations clearly reduce the number of steps needed for convergence.

Booth function.

We also test symmetry teleportation on the Booth function defined in Eqn. equation 4.4. We initialize the parameters x_1, x_2 to $(5, -5)$. Each algorithm is run 10 steps with learning rate 0.08. We perform symmetry teleportation on the parameters once, before epoch 5. The group elements are found by gradient ascent on θ for 10 steps with learning rate 0.001. θ is initialized uniformly at random over $[0, \pi)$. Similar to the Rosenbrock function, teleportation moves the parameters to a trajectory with a larger convergence rate (Figure 4.3 bottom row).

We compare the gradient at different loss values for gradient descent with and without teleportation. Figure 4.4 shows that the trajectory with teleportation has a larger dL/dt value than the trajectory without teleportation at the same loss values. Therefore, the rate of change in the loss is larger in the trajectory with teleportation, which makes it favorable.

Multilayer neural network regression.

We further evaluated our method on a three-layer neural network with a regression loss $\min_{W_1,W_2,W_3} \|Y - W_3\sigma(W_2\sigma(W_1X))\|_2$. The dimension of weight matrices are $W_3 \in \mathbb{R}^{8\times 7}$, $W_2 \in \mathbb{R}^{7\times 6}$, and $W_1 \in \mathbb{R}^{6\times 5}$. $X \in \mathbb{R}^{5\times 4}$ is the data, $Y \in \mathbb{R}^{8\times 4}$ is the target, and σ is the LeakyReLU activation with slope coefficient 0.1. Data X, Y and initialization of parameters W are set uniformly at random over $[0, 1]$. GD uses learning rate 10^{-4} and AdaGrad uses 10^{-1}. Each algorithm is run 300 steps. When using teleportation, we perform symmetry transform on the

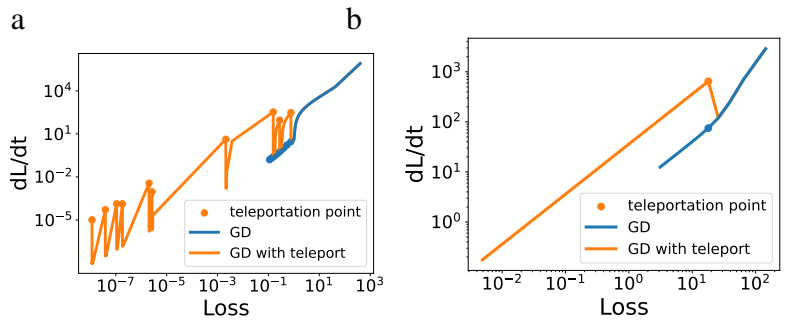


Figure 4.4. Gradient on the trajectory of optimizing the Rosenbrock function (left) and Booth function (right). At the same loss value, the graident is larger on the trajectory with teleportation, indicating a better descent path.

parameters once at epoch 5. In GD, the group elements used for these transforms are found by gradient ascent on T for 8 steps, with learning rate 10^{-7}. In AdaGrad, the group elements are found by gradient ascent for 2 steps, with learning rate 10^{-5}. The choice of hyperparameters comes from a grid search described in the next section.

Teleportation of the $\mathrm{GL}(\mathbb{R})$ group can be performed by finding an element x in its Lie algebra, such that transforming $\boldsymbol{w}$ by the group element $g = \exp(x)$ improves the gradient $\frac{dL}{dt}$. We use the first order approximation of the exponential map. Between each pair of weight matrices, we replace g_m by $I+T$ and g_m^{-1} by $I-T$ in equation 4.13, where $T \in \mathbb{R}^{d_m \times d_m}$ is initialized to 0. Then we perform gradient ascent steps on T with objective defined in Line 3 of Algorithm 1 and update the pair of weights.

Figure 4.5a and 4.5b show the training curves plotted against epochs and time. Shaded area denotes one standard deviation from 5 runs. Since GD and AdaGrad use different learning rates, they are not directly comparable. However, the addition of teleportation clearly improves both algorithms. Figure 4.5c and 4.5d shows the squared norm of gradient from a single run. Teleportation increases the magnitude of gradient, and the trajectory with teleportation has a larger dL/dt value at the same loss values, which demonstrates that teleportation finds a better trajectory.

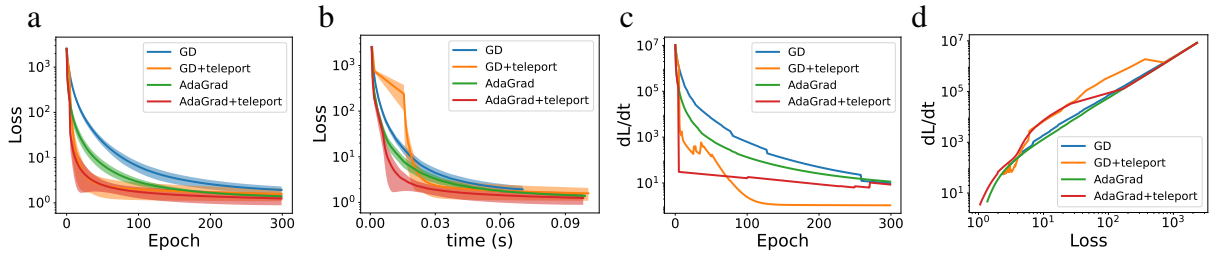


Figure 4.5. Multi-layer network optimization using gradient descent with and without teleportation. Teleportation reduces both the number of epochs and the total computational time required to reach convergence. At the same loss values, the teleported version has a larger gradient.

MNIST classification.

We apply symmetry teleportation on the MNIST classification task [34]. We split the training set into 48,000 for training and 12,000 for validation. The input data has dimension 28×28 and is flattened into a vector. The output of the neural network has dimension 10 corresponding to the 10 digit classes. We used a three-layer neural network with hidden dimension $[512, 512]$, LeakyReLU activations, and cross-entropy loss. Learning rate is 2×10^{-3}, and learning rate for teleportation is 10^{-3}. Each optimization algorithm is run 80 epochs with batch size of 20. Immediately after the first epoch, we apply teleportation using data from one mini-batch, and repeat for 4 different mini-batches. For each mini-batch, 10 gradient ascent steps are used to optimize g_m.

Figure 4.6a,b shows the effect of teleportation on training and validation loss, and Figure 4.6c,d shows the norm of the gradient in training. While teleportation significantly accelerates the decrease of the training loss in SGD, its effect on the validation loss is limited and detrimental for AdaGrad. Therefore, teleportation on MNIST makes training faster at the beginning but leads to earlier overfit and slightly worse validation accuracy. A possible reason is that regions with large gradients have sharp minima that do not generalize well.

4.6.2 Hyperparameter Tuning

To observe the effect of hyperparameters on the speedup in computation time, we did a hyperparameter sweep on the number of steps and the learning rate used in each teleporta-

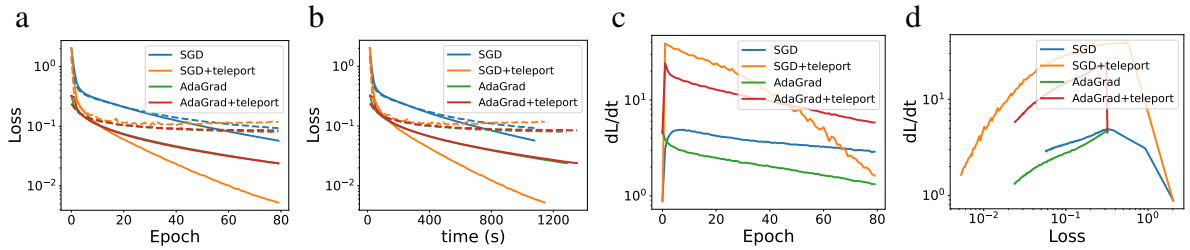


Figure 4.6. MNIST classification using gradient descent with and without teleportation. Solid lines are training loss and dashed lines are validation loss.

tion. The speedup of teleportation on SGD and AdaGrad is defined by $t_{sgd}/t_{sgd+teleport}$ and $t_{adagrad}/t_{adagrad+teleport}$ respectively, where $t_{sgd}, t_{adagrad}$ are the wall-clock time required to reach convergence using SGD or AdaGrad, and $t_{sgd+teleport}, t_{adagrad+teleport}$ are convergence time with teleportation. We consider the optimization algorithm converged if the difference between the loss of two consecutive steps is less than 10^{-3}. This experiment is run on one CPU.

Figure 4.7 shows the speedup of the same multilayer neural network regression problem defined in Section 4.6.1, but teleporting only once at epoch 5. We did a grid search for teleportation learning rates in $[10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}]$ and number of teleportation steps in $[1, 2, 4, 8, 16, 32]$. Omitted points in the figure indicate that the gradient descent fails to converge within 2000 steps or diverges.

When converged, most hyperparameter combinations improve the convergence speed in wall-clock time (speedup > 1). There are trade-offs in both the number of steps used for each teleportation and the teleportation learning rate. Increasing the number of steps used to optimize teleportation target allows us to find a better point in the parameter space but increases the cost of one teleportation. Increasing the learning rate of optimizing the group element improves $\|\partial L/\partial \boldsymbol{w}\|$ but is more likely to lead to divergence since $\|\partial L/\partial \boldsymbol{w}\|$ can become too large for the gradient descent learning rate.

4.6.3 Teleportation Schedule

The effect of teleportation varies depending on its time and frequency. Figure 4.8 shows the result of teleportation on MNIST with different hyperparameters. In all experiments, we

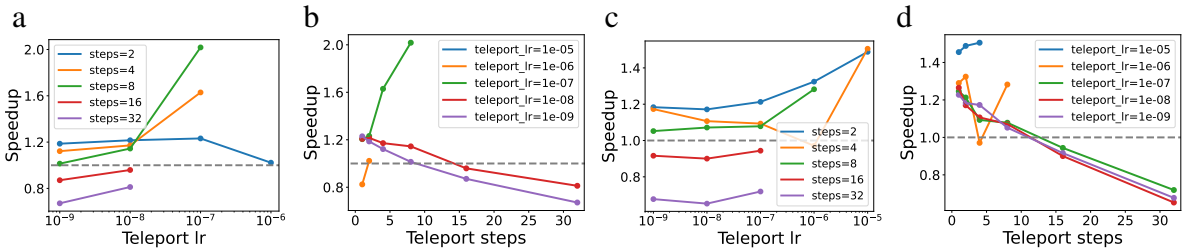


Figure 4.7. Hyperparameter sweeps of the number of steps and the learning rate used to find the optimal group element in teleportation. The wall-clock speedup of applying teleportation is shown separately for gradient descent (a)(b) and AdaGrad (c)(d). The dashed line represents speedup $= 1$.

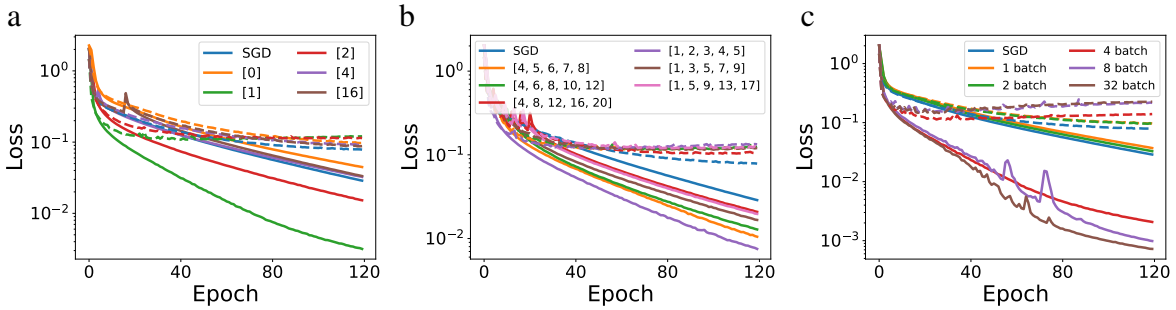


Figure 4.8. Teleportation (a) once at different epoch, (b) 5 times with different teleportation schedules, and (c) using different number of mini-batches. The lists in the legend of (a) and (c) denote the epoch numbers in teleportation schedule where teleportation happens.

use SGD with batch size 20, learning rate 2×10^{-3}, and 10 gradient ascent steps for each teleportation.

In Figure 4.8a, we randomly select 4 different mini-batches and apply teleportation on each of them individually, but at different epochs. Teleportation before training has the worst performance. After epoch 0, the effect of teleportation is stronger when it is applied earlier. In Figure 4.8b, we again apply teleportations on 4 randomly selected mini-batches, but repeat this with 5 different teleportation schedules (hyperparameter K in Algorithm 1) as shown in the legend. All schedules have the same number of teleportations. Smaller intervals between teleportations accelerate convergence more significantly. In Figure 4.8c, we apply teleportation immediately after the first epoch, but use different numbers of mini-batches and teleport using each of them individually. Using more mini-batches to teleport leads to faster decrease in training loss but is also more prone to overfitting.

4.6.4 Runtime Analysis

The additional amount of computation introduced by teleportation depends on the implementation of Line 2-5 of Algorithm 1. We discuss our implementation for teleporting multi-layer neural networks as an example. Teleportating each pair of adjacent weight matrices requires computing the inverse of the output from the previous layer. Denote the batch size as n, the largest dimension of all layers as d_{max}, and the number of layers as l. Assume that $d_{max} > n$. Computing the inverse of the output of each layer has complexity $O(d_{max}^2 n)$, and computing the pseudoinverse for all layers has complexity $O(d_{max}^2 nl)$. Note that all matrices we invert have dimensions at most $d_{max} \times n$.

For one gradient ascent step on g, the forward and backward pass both have complexity $O(d_{max}^2 nl)$. This is the same as the forward and backward pass of gradient descent on $\boldsymbol{w}$ because the architecture is the same except with approximately twice as many layers. We perform t gradient ascent steps on g. Therefore, the computation cost for one teleportation is $O(d_{max}^2 nlt)$.

We show empirically that the runtime for teleportation scales polynomially with matrix dimensions and linearly with the number of layers. We record the runtime of gradient descent on a Leaky-ReLU neural network for 300 epochs, with a 10-step teleportation every 10 epochs. Each pair of adjacent weight matrices is transformed by a group action during teleportation. Figure 4.9(a) shows the runtime for a 3-layer network using square weights and data matrices with different dimensions. Figure 4.9(b) shows the runtime for 128-by-128 weight and data matrices with different number of layers.

Although the teleportation step has the same complexity as a gradient descent step, the runtime is dominated by teleportation due to larger constants in the complexity analysis. The trade-off between teleportation time and convergence rate depends on specific problems. In our experiments, the convergence rate is improved by a small number of teleportation steps which does not add significant computational overhead.

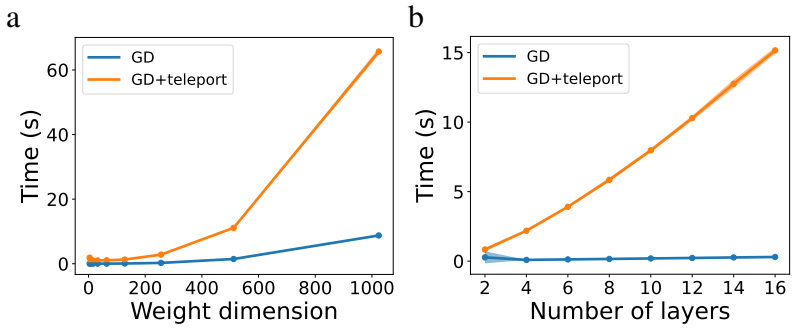


Figure 4.9. Gradient descent training time on a Leaky-ReLU neural network for 300 epochs, with a 10-step teleportation every 10 epochs. (a) 3-layer network using square weights and data matrices, with different dimensions. (b) Weights and data are all 128 by 128 matrices, but the network has different number of layers.

4.7 Discussion and Conclusions

We proposed a new optimization algorithm that exploits the symmetry in the loss landscape to improve convergence. It is interesting to note that optimizing dL/dt locally sometimes leads to an improved global path. One example is the ellipse function (Figure 4.1), where teleporting once ensures that dL/dt is optimal at every L value along the trajectory. Another example is the matrix factorization problem $L(U,V) = \|Y - UV\|_2^2$. [135] shows that the convergence rate increases with the imbalance $U^TU - V^TV$. Consider the transformation $U,V \to Ug, g^{-1}V$. To optimize $dL(U,V)/dt$ locally, we would need a large $g \in \mathrm{GL}_n$, but a large g also leads to a large $U^TU - V^TV$ which is positively correlated with the overall convergence rate of the entire trajectory. Hence teleportation is guaranteed to produce a better trajectory.

A potential future direction is to derive the exact expression for how teleportation affects the loss value at a later time in gradient flow, which may lead to a closed-form solution of the optimal teleportation destination. Additionally, inspired by the landscape view of parameter space symmetry [128], teleportation using discrete (permutation) symmetries may allow us to reach a better minimum. Finally, additional theory can be developed to explain the relationship between teleportation and second-order methods under the approximations we introduced, especially to quantify the improvement on the overall convergence rate, and to derive its effect on

generalization bounds. Integrating teleportation with other advanced optimizers such as Adam and RMSprop would be another interesting future step.

Acknowledgments

This chapter, in part, is based on material published as: Zhao, Bo; Dehmamy, Nima; Walters, Robin; Yu, Rose. “Symmetry teleportation for accelerated optimization.” Advances in neural information processing systems 35 (2022): 16679-16690 and Zhao, Bo; Gower, Robert M.; Walters, Robin; Yu, Rose. “Improving Convergence and Generalization Using Parameter Symmetries.” The Twelfth International Conference on Learning Representations 2024. The dissertation author was the primary investigator and author of these papers.

Chapter 5

Symmetry Teleportation II: Improving Generalization and Other Optimizers

In Chapter 4, we saw that symmetry teleportation is a theoretically grounded method for accelerating gradient-based optimization. In this chapter, we extend the algorithm to a new objective – improving generalization after training. In particular, we show that teleporting to minima with different curvatures improves generalization, which suggests a connection between the curvature of the minimum and generalization ability. In the second part of this chapter, we show that teleportation can readily integrate into a wide range of optimization algorithms beyond vanilla gradient descent. We then further expand the scope of teleportation by exploring the feasibility to learn the teleportation destination directly. Our results showcase the versatility of teleportation and demonstrate the potential of incorporating symmetry in optimization.

5.1 Introduction

Previous applications of teleportation are limited to accelerating optimization. We now explore a different objective – improving generalization. We relate properties of minima to their generalization ability and optimize them using teleportation. We empirically verify that certain sharpness metrics are correlated with generalization [76], although teleporting towards flatter regions has negligible effects on the validation loss. Additionally, we hypothesize that generalization also depends on the curvature of minima. For fully connected networks, we

derive an explicit expression for estimating curvatures and show that teleporting towards larger curvatures improves the model's generalizability.

To demonstrate the wide applicability of parameter space symmetry, we expand teleportation to standard optimization algorithms beyond SGD, including momentum, AdaGrad, RMSProp, and Adam. Experimentally, teleportation improves the convergence speed for these algorithms. Inspired by conditional programming and optimization-based meta-learning [9], we also propose a meta-optimizer to learn where to move parameters in a loss level set. This approach avoids the computation cost of optimization on group manifolds and improves upon existing meta-learning methods that are restricted to local updates.

The convergence speedup, applications in improving generalization, and the ability to integrate with different optimizers demonstrate the potential of improving optimization using symmetry.

5.2 Sharpness, Curvatures, and Their Relation to Generalization

Teleportation was originally proposed to speedup optimization. In this section, we explore the suitability of teleportation for improving generalization, which is another important aspect of deep learning. We first review definitions of the sharpness of minima. Then, we introduce a novel notion of the curvature of minima and discuss its implications on generalization. By observing how sharpness and curvature of minima are correlated with generalization, we improve generalization by incorporating sharpness and curvature into the objective for teleportation in the next section

Related work.

The sharpness of minima has been linked to the generalization ability of models both empirically and theoretically [64, 76, 113, 35, 166], which motivates optimization methods that find flatter minima [28, 46, 85, 77]. We employ teleportation to search for flatter points along

the loss level sets. The sharpness of a minimum is often defined using properties of the Hessian of the loss function, such as the number of small eigenvalues [76, 28, 121] or the product of the top k eigenvalues [146]. Alternatively, sharpness can be characterized by the maximum loss within a neighborhood of a minimum [76, 46, 77] or approximated by the growth in the loss curve averaged over random directions [70]. The sharpness of minima does not always capture generalization [36] [8]. Some reparametrizations do not affect generalization but can lead to minima with different sharpness.

5.2.1 Sharpness of Minima

Flat minima tend to generalize well [64], typically characterized by numerous small Hessian eigenvalues. Although Hessian-based sharpness metrics are known to correlate well with generalization, they are expensive to compute and differentiate through. To use sharpness as an objective in teleportation, we consider changes in the loss averaged over random directions. Let D be a set of vectors drawn randomly from the unit sphere $\{d \in \mathbb{R}^n : ||d|| = 1\}$, and $T \subset \mathbb{R}$ a set of displacements. Then, we have the following metric [70]:

$$\text{Sharpness:} \quad \phi(\boldsymbol{w}, T, D) = \frac{1}{|T||D|} \sum_{t \in T} \sum_{d \in D} L(\boldsymbol{w} + td). \tag{5.1}$$

5.2.2 Curvature of Minima

At a minimum, the loss-invariant or flat directions are zero eigenvectors of the Hessian. The curvature along these directions does not directly affect Hessian-based sharpness metrics. However, these curvatures may affect generalization, by themselves or by correlating to the curvature along non-flat directions. Unlike the curvature of the loss (curve $L(\boldsymbol{w})$ in Figure 5.1), the curvature of the minima (curve γ) is less well studied. We provide a novel method to quantify the curvature of the minima below.

Assume that the loss function L has a G symmetry. Consider the curve $\gamma_M : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$

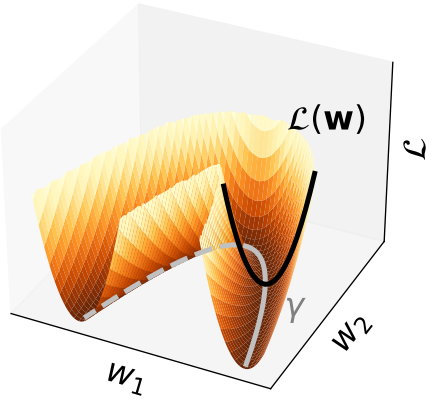


Figure 5.1. Gradient flow ($L(\boldsymbol{w})$) and a curve on the minimum (γ). The curvature of both curves may affect generalization.

where $M \in \text{Lie}(G)$ and $\gamma_M(t, \boldsymbol{w}) = \exp(tM) \cdot \boldsymbol{w}$. Then $\gamma(0, \boldsymbol{w}) = \boldsymbol{w}$, and every point on γ_M is in the minimum if $\boldsymbol{w}$ is a minimum. Let $\gamma' = \frac{d\gamma}{dt}$ be the derivative of a curve γ. The curvature of γ is $\kappa(\gamma, t) = \frac{\|T'(t)\|}{\|\gamma'(t)\|}$, where $T(t) = \frac{\gamma'(t)}{\|\gamma'(t)\|}$ is the unit tangent vector. The curvature can be written as a function of γ' and γ'', as $\kappa = \frac{\left[\|\gamma'\|^2\|\gamma''\|^2 - (\gamma' \cdot \gamma'')^2\right]^{\frac{1}{2}}}{\|\gamma'\|^3}$ [5, 125]. We assume that the action map is smooth, since calculating the curvature requires second derivatives and optimizing the curvature via gradient descent requires third derivatives.

Since the minimum can have more than one dimension, we measure the curvature of a point $\boldsymbol{w}$ on the minimum by averaging the curvature of k curves with randomly selected Lie algebra elements $M_i \in \text{Lie}(G)$. The resulting new metric is

$$\text{Curvature:} \qquad \psi(\boldsymbol{w}, k) = \frac{1}{k} \sum_{i=1}^{k} \kappa(\gamma_{M_i}(0, \boldsymbol{w}), 0) \; . \tag{5.2}$$

There are different ways to measure the curvature of a higher-dimensional manifold, such as using the Gaussian curvature of 2D subspaces of the tangent space. However, our method of approximating the mean curvature is easier to compute and suitable as a differentiable objective.

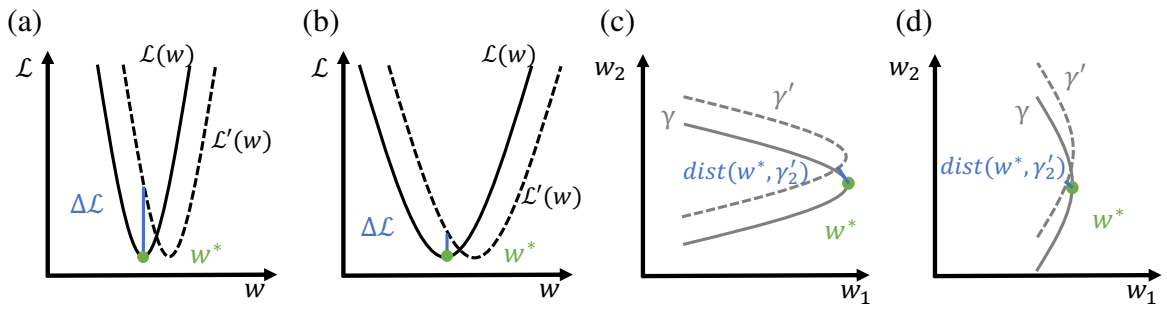


Figure 5.2. Illustration of the effect of sharpness (a,b) and curvature (c,d) of minima on generalization. See Figure 5.1 for a 3D visualization of the curves $L(\boldsymbol{w})$ and γ. When the loss landscape shifts due to a change in data distribution, sharper minima have larger increase in loss. In the example shown, minima with larger curvature moves further away from the shifted minima.

5.2.3 Correlation with Generalization

Generalization reflects how loss changes with shifts in data distribution. The sharpness of minima is well known to be correlated with generalization. Figure 5.2(a)(b) visualizes an example of the shift in loss landscape ($L(\boldsymbol{w})$), and the change of loss ΔL at a minimizer $\boldsymbol{w}^*$ is large when the minimum is sharp. The relation between the curvature of minimum and generalization is less well studied. Figure 5.2(c)(d) shows one possible shift of the minimum (γ). Under this shifting, the minimizer with a larger curvature becomes farther away from the shifted minimum. The curve on the minimum can shift in other directions.

We verify the correlation between sharpness, curvatures, and validation loss on MNIST [34], Fashion-MNIST [149], and CIFAR-10 [80]. On each dataset, we train 100 three-layer neural networks with LeakyReLU using different initializations. We generate the 100 different models used in Section 4.3 by training randomly initialized models. For all three datasets (MNIST, FashionMNIST, and CIFAR-10), we train on 50,000 samples and test on a different set of 10,000 samples. The labels for classification tasks belongs to 1 of 10 classes.

For a batch of flattened input data $X \in \mathbb{R}^{d\times 20}$ and labels $Y \in \mathbb{R}^{20}$, the loss function is $L(W_1, W_2, W_3, X, Y) = \text{CrossEntropy}\left(W_3\sigma(W_2\sigma(W_1X)), Y\right)$, where $W_3 \in \mathbb{R}^{10\times h_2}$, $W_2 \in \mathbb{R}^{h_2\times h_1}$, $W_1 \in \mathbb{R}^{h_1\times d}$ are the weight matrices, and σ is the LeakyReLU activation with slope coefficient 0.1.

For MNIST and Fashion-MNIST, $d = 28^2$, $h_1 = 16$, and $h_2 = 10$. For CIFAR-10, $d = 32^3 \times 3$, $h_1 = 128$, and $h_2 = 32$. The learning rate for stochastic gradient descent is 0.01 for MNIST and Fashion-MNIST, and 0.02 for CIFAR-10. We train each model using mini-batches of size 20 for 40 epochs.

When computing the sharpness ϕ, we choose the displacement list T that gives the highest correlation. The displacements used in this paper are $T = 0.001, 0.011, 0.021, ..., 0.191$ for MNIST, and $T = 0.001, 0.011, 0.021, ..., 0.191$ for Fashion-MNIST and CIFAR-10. We evaluate the change in loss over $|D| = 200$ random directions. For curvature ψ, we average over $k = 1$ curves generated by random Lie algebras (invertible matrices in this case).

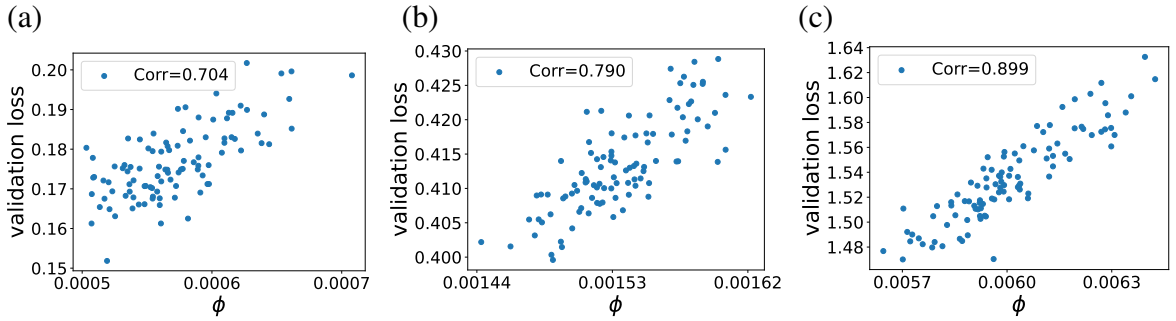


Figure 5.3. Correlation between sharpness and validation loss on MNIST (left), Fashion-MNIST (middle), and CIFAR-10 (right). Sharpness and generalization are strongly correlated.

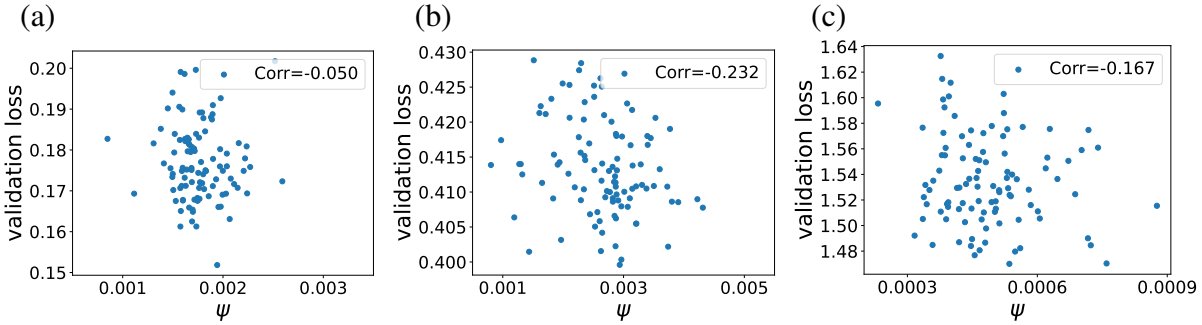


Figure 5.4. Correlation between curvature and validation loss on MNIST (left), Fashion-MNIST (middle), and CIFAR-10 (right). There is a weak negative correlation in all three datasets.

Table 5.1 shows the Pearson correlation between validation loss and sharpness or curvature. Figure 5.3 and 5.4 visualizes the correlation result in Table 5.1. Each point represents

Table 5.1. Correlation between sharpness, curvature, and validation loss.

sharpness (ϕ)			curvature (ψ)		
MNIST	Fashion-MNIST	CIFAR-10	MNIST	Fashion-MNIST	CIFAR-10
0.704	0.790	0.899	-0.050	-0.232	-0.167

one model. In all three datasets, sharpness has a strong positive correlation with validation loss, meaning that the average change in loss under perturbations is a good indicator of test performance. For the architecture we consider, the curvature of minima is negatively correlated with the validation loss. We observe that the magnitudes of the curvatures are small, which suggests that the minima are relatively flat.

5.2.4 More Intuition on Curvatures and Generalization

Example: curvature affects average displacement of minima

Consider an optimization problem with two variables $w_1, w_2 \in \mathbb{R}$. Assume that the minimum is a curve $\gamma : \mathbb{R} \to \mathbb{R}^2$ in the two-dimensional parameter space. For a point $\boldsymbol{w}_0$ on γ, we estimate its generalization ability by computing the expected distance between $\boldsymbol{w}_0$ and the new minimum obtained by shifting γ.

We consider the following two curves as examples:

$$\begin{aligned} &\gamma_1 : \mathbb{R} \to \mathbb{R}^2, t \mapsto (t, k_1 t^2) \\ &\gamma_2 : [0, 2\pi] \to \mathbb{R}^2, \theta \mapsto (k_2 \cos(\theta), k_2 \sin(\theta) + k_2), \end{aligned} \tag{5.3}$$

with $k_1, k_2 \in \mathbb{R}^{\neq 0}$. The curve γ_1 is a parabola with curvature $\kappa_1 = 2k_1$ at $\boldsymbol{w}_0 = (0,0)$. The curve γ_2 is a circle, with curvature $\kappa_2 = \frac{1}{k_2}$ at $\boldsymbol{w}_0$. Note that γ_1 is the only polynomial approximation with integer power ($\gamma(t) = (t, k|t|^n), n \in \mathbb{Z}^+$) where the curvature at $\boldsymbol{w}_0$ depends on k. When $n < 1$, the value of $\boldsymbol{w}_0$ is undefined. When $n = 1$, the first derivative at $\boldsymbol{w}_0$ is undefined. When $n > 2$, $\kappa(\boldsymbol{w}_0) = 0$.

Assume that a distribution shift in data causes γ to shift by a distance r, and that the

direction of the shift is chosen uniformly at random over all possible directions. Viewing from the perspective of the curve, this is equivalent to shifting $\boldsymbol{w}_0$ by distance r.

The distance between a point $\boldsymbol{w}$ and a curve γ is

$$dist(\boldsymbol{w}, \gamma) = \min_{\boldsymbol{w}' \in \gamma_2} \|\boldsymbol{w}' - \boldsymbol{w}\|_2. \tag{5.4}$$

Let S_r be the circle centered at the origin with radius r. The expected distance between the old solution $\boldsymbol{w}_0$ and shifted curve is

$$\mathbb{E}_{\boldsymbol{w} \in S_r}[dist(\boldsymbol{w}, \gamma)] = \frac{\int_{S_r} dist(\boldsymbol{w}, \gamma) ds}{\int_{S_r} ds} = \frac{\int_0^{2\pi} dist((r\cos\theta, r\sin\theta), \gamma) r d\theta}{\int_0^{2\pi} r d\theta}. \tag{5.5}$$

In the limit of zero curvature, γ is a straight line $\gamma(t) = (t, 0)$. In this case, the expected distance is

$$\mathbb{E}_{\boldsymbol{w} \in S_r}[dist(\boldsymbol{w}, \gamma)] = \frac{\int_0^{2\pi} |r\sin\theta| r d\theta}{2\pi r} = \frac{2r}{\pi} \approx 0.637r. \tag{5.6}$$

Figure 5.5(b)(c) shows that the expected distance's dependence on κ. Using both curves γ_1 and γ_2, the generalization ability of $\boldsymbol{w}_0$ depends on the curvature at $\boldsymbol{w}_0$. However, the type of dependence is affected by the type of curve used. In other words, the curvatures at points around $\boldsymbol{w}_0$ affect how the curvature at $\boldsymbol{w}_0$ affects generalization. Therefore, from these results alone, one cannot deduce whether minima with sharper curvatures generalize better or worse. To find a more definitive relationship between curvature and generalization, further investigation on the type of curves on the minimum is required.

We emphasize that this example only serves as an intuition for connecting curvature to generalization. As a future direction, it would be interesting to consider different families of parametric curves, higher dimensional parameter spaces, and deforming in addition to shifting the minima.

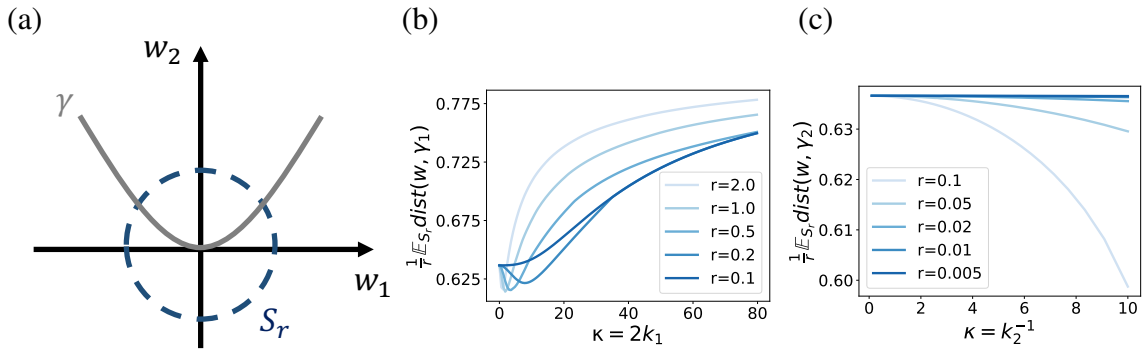


Figure 5.5. (a) Illustration of the parameter space, the minimum (γ), and all shifts with distance r (S_r). (b) Expected distance between $\boldsymbol{w}_0$ and the new minimum as a function of κ, for quadratic approximation γ_1. (c) Expected distance between $\boldsymbol{w}_0$ and the new minimum as a function of κ, for constant curvature approximation γ_2. The expected distance is scaled by r so that the curves can be plotted together.

Higher dimensions

Figure 5.6 visualizes a curve obtained from a 2D minimum. However, it is not immediately clear what curves look like on a higher-dimensional minimum. A possible way to extend previous analysis is to consider sectional curvatures.

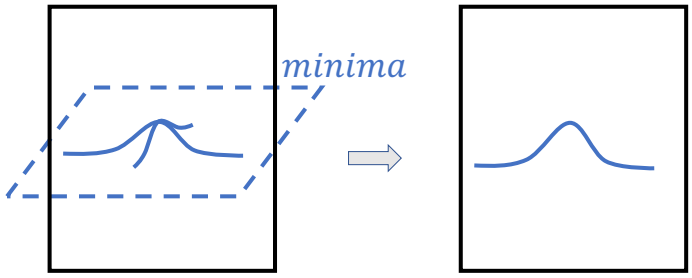


Figure 5.6. Left: a 2D minima in a 3D parameter space. Right: a 2D subspace of the parameter space and a curve on the minima (the intersection of the minima and the subspace).

5.3 Teleportation for Improving Generalization

To improve the generalization ability of the minimizer and to gain understanding of the curvature of minima, we teleport parameters to regions with different sharpness and curvature.

Multi-layer neural networks have $GL(\mathbb{R})$ symmetry between layers. We parametrize the group by its Lie algebra T, and perform gradient ascent on T to maximize the gradient norm at the transformed parameters $|\nabla L|_{\exp(T)\cdot w}|$. Algorithm 2 demonstrates how to increase curvature ψ by teleporting two layers, with hidden dimension h, in an MLP. In experiments, we use an extended version of the algorithm, which teleports all layers by optimizing on a list of T's concurrently. During teleportation, we perform gradient descent on the group elements to change ϕ or ψ. Results are averaged over 5 runs.

Algorithm 2: Changing curvature using teleportation

Input: loss function $L(w)$, parameters before teleportation w_0, teleportation learning rate $\eta_{teleport}$, number of teleportation steps $n_{teleport}$.
Output: parameters after teleportation $w_{n_{teleport}}$.
for $t = 0$ **to** $n_{teleport} - 1$ **do**
    initialize $T = 0_{h\times h}$
1   set $w'_t = (I_{h\times h} + T)\cdot w_t$
    compute $grad = \frac{d|\psi(w'_t)|}{dT}$
    set $T_t = \eta_{teleport} \times grad$
    set $w_{t+1} = (I + T_t)\cdot w_t$
end for
Return $w_{n_{teleport}}$

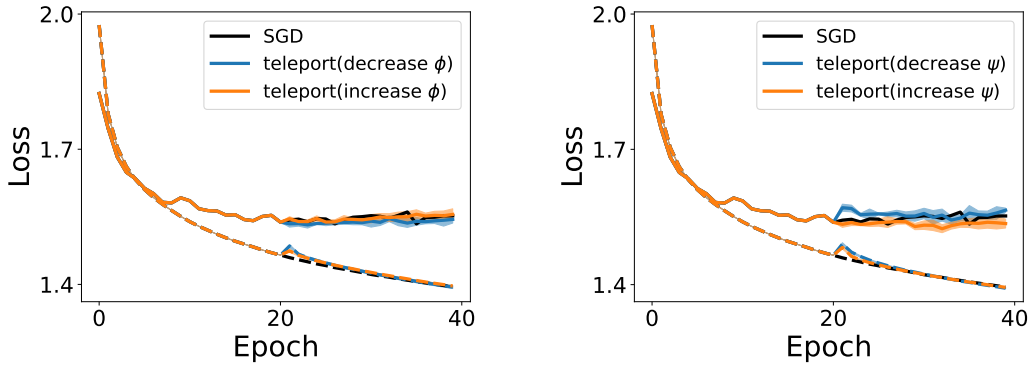


Figure 5.7. Changing sharpness (left) or curvature (right) using teleportation and its effect on generalization on CIFAR-10. Solid line represents average test loss, and dashed line represent average training loss. Teleporting to decrease sharpness improves validation loss slightly. Teleportation changing curvatures has a more significant impact on generalization ability.

On CIFAR-10, we run SGD using the same three-layer architecture, but with a smaller

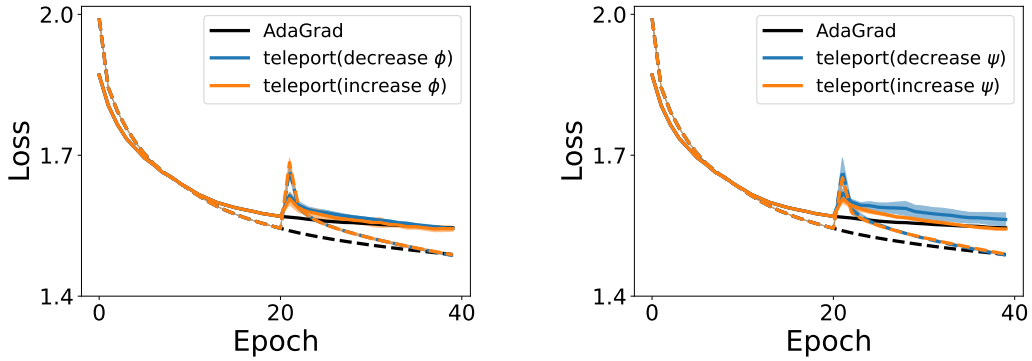


Figure 5.8. Changing sharpness (left) or curvature (right) using teleportation and its effect on generalizability of AdaGrad solutions on CIFAR-10. Solid line represents average test loss, and dashed line represent average training loss.

hidden size $h_1 = 32$ and $h_2 = 10$. At epoch 20 which is close to convergence, we teleport using 5 batches of data, each of size 2000. During each teleportation for ϕ, we perform 10 gradient ascent (or descent) steps on the group element. During each teleportation for ψ, we perform 1 gradient ascent (or descent) step on the group element. The learning rate for the optimization on group elements is 5×10^{-2}.

Figure 5.7 shows the training curve of SGD on CIFAR-10, with one teleportation at epoch 20. We observer similar results for AdaGrad (Figure 5.8). Teleporting to flatter points slightly improves the validation loss, while teleporting to sharper points has no effect. Since the group action keeps the loss invariant only on the batch of data used in teleportation, the errors incurred in teleportation have a similar effect to a warm restart, which makes the effect of changing sharpness less clear.

Interestingly, by changing the curvature, teleportation is able to affect generalization. Teleporting to points with larger curvatures helps find a minimum with lower validation loss, while teleporting to points with smaller curvatures has the opposite effect. This suggests that at least locally, curvature is correlated with generalization.

5.4 Applications to Other Optimization Algorithms

Having shown teleportation's potential to improve optimization and generalization, we demonstrate its wide applicability by integrating teleportation into different optimizers and meta-learning.

5.4.1 Integrating Teleportation with Momentum and AdaGrad

Setup.

We test teleportation with various algorithms using the a 3-layer neural network and mean square error: $\min_{W_1,W_2,W_3} \|Y - W_3\sigma(W_2\sigma(W_1X))\|_2$, with data $X \in \mathbb{R}^{5\times 4}$, target $Y \in \mathbb{R}^{8\times 4}$, and weight matrices $W_3 \in \mathbb{R}^{8\times 7}$, $W_2 \in \mathbb{R}^{7\times 6}$, and $W_1 \in \mathbb{R}^{6\times 5}$. The activation function σ is LeakyReLU with slope coefficient 0.1. Each element in the weight matrices is initialized uniformly at random over $[0,1]$. Data X, Y are randomly generated also from $[0,1]$.

Momentum.

We compare three strategies of integrating teleportation with momentum: teleporting both parameters and momentum, teleporting parameters but not momentum, and reset momentum to 0 after a teleportation. In each run, we teleport once at epoch 5. Each strategy is repeated 5 times.

The training curves of teleporting momentum in different ways are similar (Figure 5.9a), possibly because the momentum accumulated is small compared to the gradient right after teleportations. All methods of teleporting momentum improves convergence, which means teleportation works well with momentum.

AdaGrad.

In AdaGrad, the rate of change in loss is

$$\frac{dL(\boldsymbol{w})}{dt} = \frac{\partial L}{\partial \boldsymbol{w}}^T \frac{d\boldsymbol{w}}{dt} = -\eta \|\nabla L\|_A, \tag{5.7}$$

where $\eta \in \mathbb{R}$ is the learning rate and $\|\nabla L\|_A$ the Mahalanobis norm with $A = (\varepsilon I + diag(G_{t+1}))^{-\frac{1}{2}}$. Previously, we optimize $\|\nabla L\|_2$ in teleportation. We compare that to optimizing $\|\nabla L\|_A$. Since the magnitude of A is different than 1, a different learning rate for the gradient ascent in teleportation is required. We choose the largest learning rate (with two significant figures) that does not lead to divergence. The teleportation learning rates used are 1.2×10^{-5} for objective $\max_g \|\nabla L\|_2$ and 7.5×10^{-3} for objective $\max_g \|\nabla L\|_A$.

Teleporting using the group element that optimizes $\|\nabla L\|_A$ has a slight advantage (Figure 5.9b). Similar to the observations in [157], teleportation can be integrated into adaptive gradient descents.

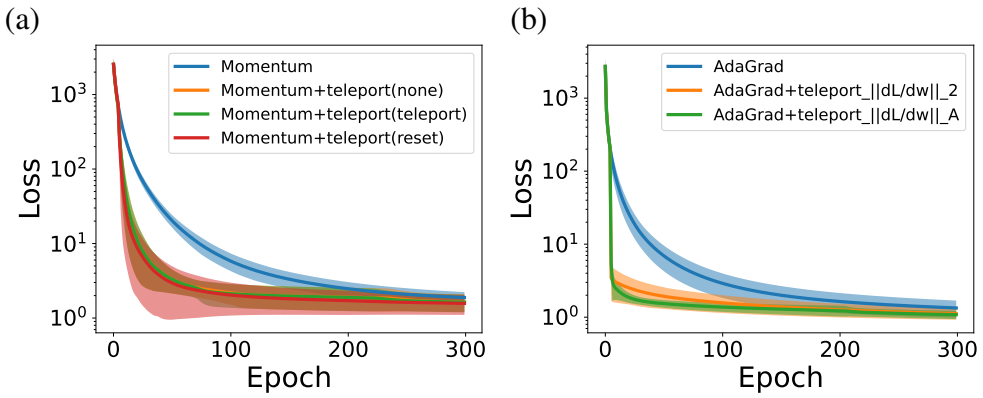


Figure 5.9. Comparison of different methods of integrating teleportation with momentum and AdaGrad.

5.4.2 Standard Optimizers

Teleportation improves optimization not only for SGD. To show that teleportation works well with other standard optimizers, we train a 3-layer neural network on MNIST using different optimizers with and without teleportation.

During training, we teleport once at the first epoch, using 8 minibatches of size 200. We use a three-layer model and cross-entropy loss for classification with minibatches of size 20. For a batch of flattened input data $X \in \mathbb{R}^{28^2 \times 20}$ and labels $Y \in \mathbb{R}^{20}$, the loss function is $L(W_1, W_2, W_3, X, Y) = \text{CrossEntropy}\,(W_3\sigma(W_2\sigma(W_1X)), Y)$, where $W_3 \in \mathbb{R}^{10\times 10}$, $W_2 \in \mathbb{R}^{10\times 16}$,

$W_1 \in \mathbb{R}^{16\times 28^2}$ are the weight matrices, and σ is the LeakyReLU activation with slope coefficient 0.1. The learning rates are 10^{-4} for AdaGrad, and 5×10^{-2} for SGD with momentum, RMSProp, and Adam. The learning rate for optimizing the group element in teleportation is 5×10^{-2}, and we perform 10 gradient ascent steps when teleporting using each mini-batch. We use 50,000 samples from training set for training, and 10,000 samples in the test set for testing.

Figure 5.10 shows that teleporation improves the convergence of Adagrad. As for SGD with momentum, RMSProp, and Adam, there is only an improvement in the first 10 epoches. The runtime for a teleportation is smaller than the time required to train one epoch, hence teleportation improves convergence rate per epoch at almost no additional cost of time (Figure 5.11).

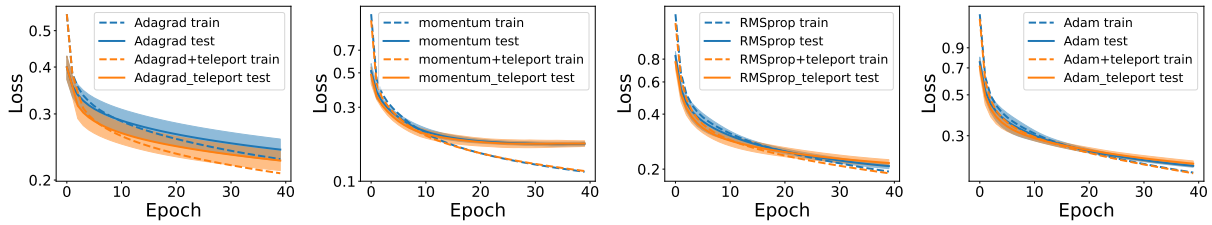


Figure 5.10. Integrating teleportation with AdaGrad, momentum, RMSProp, and Adam improves the convergence rate on MNIST. Solid line represents the average test loss, and dashed line represents the average training loss. Shaded areas are 1 standard deviation of the test loss across 5 runs.

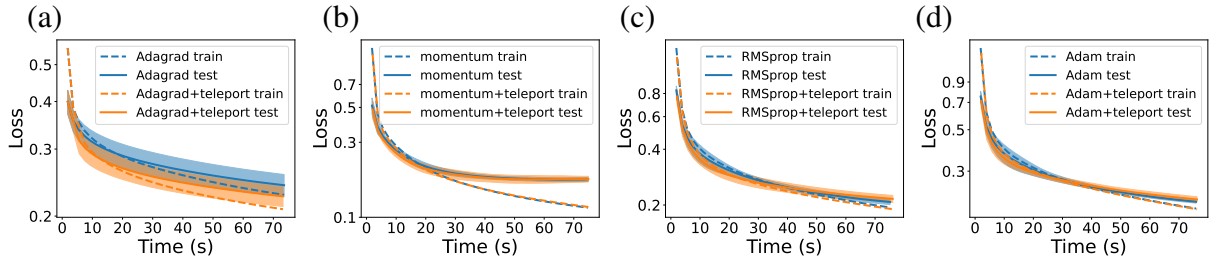


Figure 5.11. Runtime comparison for integrating teleportation into various algorithms. Solid line represents average training loss, and dashed line represents average test loss. Shaded areas are 1 standard deviation of the test loss across 5 runs. The plots look almost identical to Figure 5.10, indicating that the cost of teleportation is negligible compared to gradient descents.

5.5 Learning to Teleport

In optimization-based meta-learning, the parameter update rule or the hyperparameters are learned using a meta-optimizer [9, 44]. Teleportation introduces an additional degree of freedom in parameter updates. We augment existing meta-learning algorithms by learning both the local update and teleportation. This allows us to teleport without implementing the additional optimization step on groups, which reduces computation time.

Let $\boldsymbol{w}_t \in \mathbb{R}^d$ be the parameters at time t, and $\nabla_t = \frac{\partial L}{\partial \boldsymbol{w}}\Big|_{\boldsymbol{w}_t}$ be the gradient of the loss L. In gradient descent, the update rule with learning rate η is

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla_t.$$

In meta-learning [9], the update on $\boldsymbol{w}_t$ is learned using a meta-learning optimizer m, which takes ∇_t as input. Here m is an LSTM model. Denote h_t as the hidden state in the LSTM and ϕ as the parameters in m. The update rule is

$$\begin{gathered} \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + f_t \\ \begin{bmatrix} f_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi). \end{gathered}$$

Extending this approach beyond an additive update rule, we learn to teleport. Let G be a group whose action on the parameter space leaves L invariant. We use two meta-learning optimizers m_1, m_2 to learn the update direction $f_t \in \mathbb{R}^d$ and the group element $g_t \in G$:

$$\begin{gathered} \boldsymbol{w}_{t+1} = g_t \cdot (\boldsymbol{w}_t + f_t) \\ \begin{bmatrix} f_t \\ h_{1_{t+1}} \end{bmatrix} = m_1(\nabla_t, h_{1_t}, \phi_1), \quad \begin{bmatrix} g_t \\ h_{2_{t+1}} \end{bmatrix} = m_2(\nabla_t, h_{2_t}, \phi_2). \end{gathered}$$

Experiment setup.

We train and test on two-layer neural networks $L(W_1, W_2) = \|Y - W_2\sigma(W_1X)\|_2$, where $W_2, W_1, X, Y \in \mathbb{R}^{20\times 20}$, and σ is the LeakyReLU function with slope coefficient 0.1. Both meta-optimizers are two-layer LSTMs with hidden dimension 300. We train the meta-optimizers on multiple trajectories created with different initializations, each consisting of 100 steps of gradient descent on L with random X, Y and randomly initialized W's. We update the parameters in m_1 and m_2 by unrolling every 10 steps. The learning rate for meta-optimizers are 10^{-4} for m_1 and 10^{-3} for m_2. We test the meta-optimizers using 5 trajectories not seen in training.

Algorithm 3 summarizes the training procedure. The vanilla gradient descent baseline ("GD") uses the largest learning rate that does not lead to divergence (3×10^{-4}). The second baseline ("LSTM(update)") learns the update f_t only and does not perform teleportation ($g_t = I, \forall t$). The third baseline ("LSTM(lr,tele)") learns the group element g_t and the learning rate used to perform gradient descent instead of the update f_t. We keep training until adding more training trajectories does not improve convergence rate. We use 700 training trajectories for our approach, 600 for the second baseline, and 30 for the third baseline.

Results. By learning both the local update f_t and non-local transformation g_t, our meta-optimizer successfully learns to learn faster. Figure 5.12 shows the improvement of our approach from the previous meta-learning method, which only learns f_t. Compared to the baselines, learning the two types of updates together ("LSTM(update,tele)") achieves better convergence rate than learning them separately. Additionally, learning the group element g_t eliminates the need for performing gradient ascent on the group manifold and reduces hyperparameter tuning for teleportation. As an example of successful integration of teleportation into existing optimization algorithms, this toy experiment demonstrates the flexibility and promising applications of teleportation.

Algorithm 3: Learning to teleport

Input: Loss function L, learning rate η, number of epochs T, LSTM models m_1, m_2 with initial parameters ϕ_1, ϕ_2, unroll step t_{unroll}.
Output: Trained parameters ϕ_1 and ϕ_2.
for each training initialization **do**
  for $t = 1$ **to** T **do**
    $f_t, h_{1_{t+1}} = m_1(\nabla_t, h_{1_t}, \phi_1)$
1   $g_t, h_{2_{t+1}} = m_2(\nabla_t, h_{2_t}, \phi_2)$
    $\boldsymbol{w} \leftarrow g_t \cdot (\boldsymbol{w} + f_t)$
    if $t \mod t_{unroll} = 0$ **then**
      update ϕ_1, ϕ_2 by back-propogation from accumulated loss $\sum_{i=t-t_{unroll}}^{t} L(\boldsymbol{w}_i)$
    end if
  end for
end for

5.6 Discussion

Teleportation is a powerful tool to search in the loss level sets for parameters with desired properties. We provide theoretical guarantees that teleportation accelerates the convergence rate of SGD. Using concepts in symmetry, we propose a novel notion of curvature and show that incorporating additional teleportation objectives such as changing the curvatures can be beneficial to generalization. The close relationship between symmetry and optimization opens up a number of exciting opportunities. Exploring other objectives in teleportation appears to be an interesting future direction. Other possible applications include extending teleportation to different architectures, such as convolutional or graph neural networks, and to different algorithms, such as sampling-based optimization.

The empirical results linking sharpness and curvatures to generalization are intriguing. However, the theoretical origin of their relation remains unclear. In particular, a precise description of how the loss landscape changes under distribution shifts is not known. More investigation of the correlation between curvatures and generalization will help teleportation to further improve generalization and take us a step closer to understanding the loss landscape.

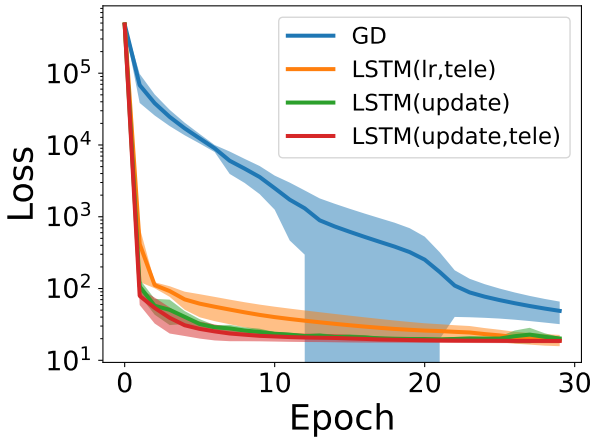


Figure 5.12. Performance of the trained meta-optimizer on the test set. Learning both local update f_t and nonlocal transformation g_t results in better convergence rate than learning only local updates or learning only teleportation.

Acknowledgement

This chapter, in part, is based on material published as: Zhao, Bo; Gower, Robert M.; Walters, Robin; Yu, Rose. "Improving Convergence and Generalization Using Parameter Symmetries." The Twelfth International Conference on Learning Representations 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 6

An Empirical Approach for Discovering Parameter Space Symmetry

In Chapters 2 through 5, we saw that parameter space symmetry is important for understanding neural networks' loss landscape, training dynamics, and generalization. However, identifying the full set of these symmetries remains a challenge. In this chapter, we explore an empirical approach for finding symmetries. We formalize data-dependent parameter symmetries and derive their infinitesimal form, which enables an automated approach to discovering symmetry across different architectures. Our framework systematically uncovers parameter symmetries, including previously unknown ones. We also prove that symmetries in smaller subnetworks can extend to larger networks, enabling direct generalization of discovered symmetries to more complex models.

6.1 Introduction

Parameter space symmetry, or loss-invariant transformation of parameters, influences various aspects of deep learning theory. Continuous symmetry connects groups to their orbits, revealing topological properties such as the dimension [160] and connectedness [159] of the minimum. Parameter symmetry also influences training dynamics through the associated conserved quantities of gradient flow [82] and by steering stochastic gradient descent towards certain favored solutions [168]. Additionally, symmetry provides a tool to perform optimization within a

loss level set, with successful applications in accelerating optimization [11, 157] and improving generalization [161]. Other applications of parameter symmetry include model compression [51, 132], efficient sampling in Bayesian neural networks [144], and equivariant architectures for weight space learning [106, 165].

Despite the wide range of applications, our knowledge of parameter space symmetries remains limited. In particular, known symmetries often cannot account for all loss-invariant parameter transformations. While several frameworks have been developed to unify known symmetries, whether the symmetries in current literature are complete remains an open question. The lack of a systematic approach necessitates deriving symmetries from scratch for each new architecture, creating barriers for broader application of parameter symmetries.

In this paper, we present an automated approach for discovering symmetry groups and their group actions in the parameter space of neural networks. To define the search space, we formalize data-dependent symmetries and derive their infinitesimal version, which simplifies the discovery architecture. Additionally, we learn the action maps directly using a neural network, enabling the discovery of nonlinear group actions. By including data-dependent and nonlinear group actions, our framework is capable of capturing a broader range of symmetries than previously considered.

While directly searching for symmetries in modern architectures with billions of parameters is prohibitively expensive, we show that large networks often inherit symmetries from their components or subnetworks. Analyzing these smaller networks provides an efficient and scalable way to uncover many symmetries in larger architectures. By extending the symmetries of small networks to their larger counterparts, our method sidesteps the complexity of handling high-dimensional parameter spaces, reducing computational cost. This approach also provides a framework for leveraging small-scale symmetries to better understand the structure of more complex architectures.

In summary, our main contributions are:

- A formal definition of data-dependent parameter symmetries and their infinitesimal form.
- An approach to identify symmetries in the parameter space of large networks from known symmetries in smaller subnetworks.
- A framework for automated discovery of parameter symmetries across several neural architectures.
- Evidence of previously unknown symmetries that are data-dependent or act on non-contiguous layers.

6.2 Related Work

Parameter space symmetry.

Parameter symmetries are loss-invariant transformations on neural network parameters, often in the form of group actions. Symmetry often arises from equivariance of common activation functions [55], including invertible linear transformations in linear networks, rescaling in homogeneous networks [16, 38, 114], radial rescaling in radial neural networks [51], and translation in softmax and scaling in batchnorm functions [82]. In tanh networks [29], only permutation and sign flip symmetries preserve the loss function. ReLU networks, however, possess symmetries beyond the well-known rescaling [57]. The existence and number of symmetries in most other architectures remain as open questions.

Data-dependent symmetry.

While the above symmetries leave the loss unchanged on all data, a relaxed definition, data-dependent symmetry, only requires loss invariance on a subset of data. [160] found examples of such symmetries with nontrivial data dependency, although these symmetries are complicated, limited to minibatches of size one, and difficult to generalize across different architectures. This motivates an automated symmetry discovery framework, which, in principle, can find symmetries of arbitrary form in arbitrary architectures. The concept of a symmetry dependent on data has

also appeared in adjacent fields. For example, [103] observe that learned data invariance in neural networks is strongly conditioned on data and breaks under data distribution drift; [131] define a joint group action on data and parameters as part of a new proof of universal approximation theory.

Discovering and measuring symmetry.

Various work explores learning continuous symmetries by identifying generators of Lie groups [79, 104, 33, 152, 50], including cases with nonlinear group actions [151, 124]. We build on this approach to discover data-dependent group action in high-dimensional parameter spaces. While learning discrete symmetry [164, 75] and distributions of symmetry [20, 119, 138] are also relevant, they are not the primary focus of this paper.

Extracted symmetry is often evaluated locally, by measuring function changes under infinitesimal symmetry transformations [59] or by comparing tangent spaces of orbits under the learned group and the true symmetry group [116]. We adopt the local invariance of loss functions under symmetry transformation, similar to that defined in [59, 104], as the minimization objective in learning data-dependent group actions.

6.3 Infinitesimal Data-Dependent Symmetry

In this section, we derive an alternative definition for data-dependent symmetries using Lie algebras. For the automatic symmetry discovery framework in Section 6.5, these definitions allow us to learn the group elements and actions without computing the matrix exponential, which is expensive, during training. We also provide examples of symmetries in common neural networks.

6.3.1 Definitions

Let Param be the space of parameters and $\mathscr{D}$ be the space of data. In this paper, we consider loss functions of the form $L\colon \mathsf{Param} \times \mathscr{D} \to \mathbb{R}$, which map parameters and a single data

point to a real number. By abuse of notation, we allow L to simultaneously process multiple data points. Specifically, we sometimes define $L\colon \mathsf{Param} \times \mathscr{D}^d \to \mathbb{R}^d$ for $d \in \mathbb{N}$ data points.

In this chapter, we restrict the symmetry group G to be a linear group. That is, we assume there is a faithful representation $\rho\colon G \to \mathrm{GL}(n)$. The corresponding Lie algebra representation $d\rho\colon \mathfrak{g} \to \mathfrak{gl}(n)$ is the differential of ρ, mapping elements of the Lie algebra $\mathfrak{g}$ of G to the Lie algebra $\mathfrak{gl}(n)$ of $GL(n)$. If G is a subgroup of $\mathrm{GL}(n)$, then ρ is the inclusion map, and consequently, $d\rho$ is the inclusion of $\mathfrak{g}$ into $\mathfrak{gl}(n)$.

The following theorem shows that the derivative of the loss function L with respect to the parameters θ vanishes in the directions generated by the symmetry group's infinitesimal transformations. In other words, the loss function is invariant to small changes along these symmetric directions in parameter space.

Theorem 6.3.1 (Infinitesimal version of loss invariance)**.** *Let $a\colon \mathscr{D}^d \to (G \times \mathsf{Param} \to \mathsf{Param})$ be a parameter space symmetry of a loss function $L\colon \mathsf{Param} \times \mathscr{D}^d \to \mathbb{R}^d$. Assume that L and a_X is differentiable at all $X \in \mathscr{D}^d$. Let $D_\theta L|_{\theta,X} \in \mathbb{R}^{d \times \dim(\mathsf{Param})}$ be the derivative of L with respect to θ, and $D_g a_X|_{I,\theta} \in \mathbb{R}^{\dim(\mathsf{Param}) \times \dim \mathfrak{g}}$ be the derivative of $a_X(g,\theta)$ with respect to g. Then, for all $\theta \in \mathsf{Param}$, $X \in \mathscr{D}^d$, and $h \in \mathfrak{g}$,*

$$D_\theta L|_{\theta,X}\, D_g a_X|_{I,\theta}(h) = 0. \tag{6.1}$$

Proof sketch. Consider a smooth curve $\gamma(t) = a_X(\exp(ht),\theta)$ in Param, where $h \in \mathfrak{g}$ and $t \in \mathbb{R}$. Then, since L is invariant under a, $L(\gamma(t),X) = L(\theta,X), \forall t \in \mathbb{R}$. The result follows from differentiating both sides with respect to t at $t = 0$ and applying the chain rule. □

Proof. Since a is a symmetry of L, we have

$$L(a_X(g,\theta),X) = L(\theta,X), \quad \forall g \in G, \quad \forall \theta \in \mathsf{Param}, \quad \forall X \in \mathscr{D}^d.$$

Consider a smooth curve $\gamma(t) = a_X(\exp(ht),\theta)$ in Param, where $h \in \mathfrak{g}$ and $t \in \mathbb{R}$. Then,

since L is invariant under a,

$$L(\gamma(t),X) = L(\theta,X), \quad \forall t \in \mathbb{R}.$$

Differentiating both sides with respect to t at $t = 0$, we get

$$\frac{d}{dt} L(\gamma(t),X)\bigg|_{t=0} = 0.$$

Applying the chain rule and noting that $\gamma(t)|_{t=0} = \theta$,

$$\frac{d}{dt} L(\gamma(t),X)\bigg|_{t=0} = D_\theta L|_{\theta,X} \left(\frac{d\gamma(t)}{dt}\bigg|_{t=0} \right).$$

Now, compute $\frac{d\gamma(t)}{dt}\Big|_{t=0}$ using the chain rule:

$$\frac{d\gamma(t)}{dt}\bigg|_{t=0} = \frac{d}{dt} a_X(\exp(ht),\theta)\bigg|_{t=0} = D_g a_X|_{I,\theta} \left(\frac{d}{dt}\exp(ht)\bigg|_{t=0} \right).$$

Since exp is the exponential map from $\mathfrak{gl}(n)$ to $GL(n)$, and $h \in \mathfrak{gl}(n)$, we have

$$\frac{d}{dt}\exp(ht)\bigg|_{t=0} = h.$$

Therefore,

$$\frac{d\gamma(t)}{dt}\bigg|_{t=0} = D_g a_X|_{I,\theta}(h).$$

Putting it all together,

$$D_\theta L|_{\theta,X}\left(D_g a_X|_{I,\theta}(h)\right) = 0.$$

□

Equation equation 6.1 states that the gradient of the loss function L with respect to the

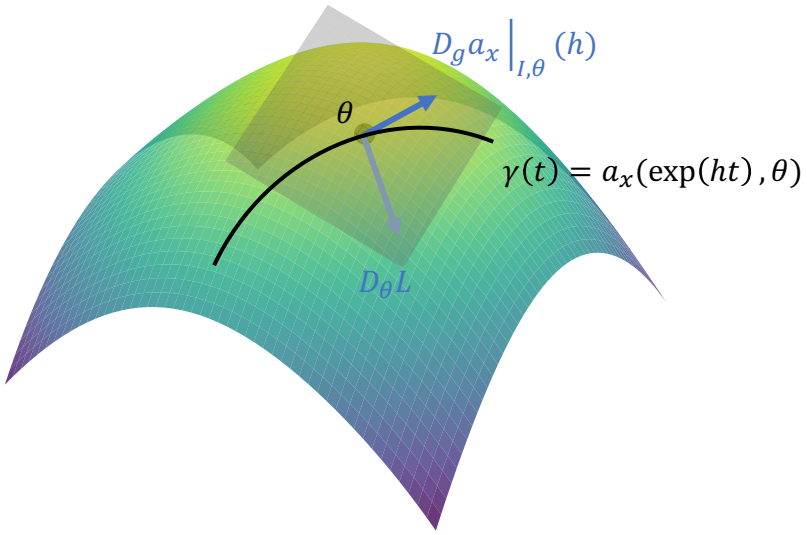


Figure 6.1. Illustration of an infinitesimal action $D_g a_X\big|_{I,\theta}(h)$ and loss gradient $D_\theta L$ at a point θ in the parameter space. The infinitesimal action is in the tangent space of the loss level set at θ.

parameters θ is orthogonal to the directions in parameter space generated by the infinitesimal action $D_g a_X\big|_{I,\theta}(h)$. This orthogonality implies that moving along these symmetric directions does not change the loss to first order, reflecting the invariance of L under the group action. Figure 6.1 illustrates relevant directions.

Similar to the infinitesimal formulation of loss invariance, we provide an infinitesimal version of the associativity law for smooth group actions on parameter space. While Theorem 6.3.1 shows how local transformations in the Lie algebra preserve the value of the loss function, associativity ensures that applying successive group transformations is consistent with their composition in the group. The theorem below shows how associativity manifests at the infinitesimal level through the derivatives of the action map a_X.

Theorem 6.3.2 (Infinitesimal version of associativity)**.** *Let G be a Lie group, and let $a\colon \mathscr{D}^d \to (G \times \mathsf{Param} \to \mathsf{Param})$ be a smooth map that satisfies the associative law. Then, for all $\theta \in \mathsf{Param}$, $X \in \mathscr{D}^d$, and $h_1, h_2 \in \mathfrak{g}$,*

$$D_\theta a_X\big|_{I,\theta}\left(D_g a_X\big|_{I,\theta}(h_1)\right) + D_g a_X\big|_{I,\theta}(h_2)$$

$$= D_g a_X\big|_{I,\theta}(h_1 + h_2).$$

Proof. To capture infinitesimal behavior, we write $g_1 = \exp(th_1)$ and $g_2 = \exp(th_2)$, where $h_1, h_2 \in \mathfrak{g}$, and $t \in \mathbb{R}$. We then take the derivative of both sides of the associativity axiom with respect to t, at $t = 0$.

For the left side:

$$\begin{aligned}\frac{d}{dt} a_X(g_2, a_X(g_1, \theta)) &= \frac{\partial}{\partial g_1} a_X(g_2, a_X(g_1, \theta)) \frac{\partial g_1}{\partial t} + \frac{\partial}{\partial g_2} a_X(g_2, a_X(g_1, \theta)) \frac{\partial g_2}{\partial t} \\ &= D_\theta a_X\big|_{I,\theta}\Big(D_g a_X\big|_{I,\theta}(h_1)\Big) + D_g a_X\big|_{I,\theta}(h_2).\end{aligned}$$

Following from the Baker-Campbell-Hausdorff formula in the first-order approximation, $g_2 g_1 \approx \exp(t(h_1 + h_2))$ for small t. For the right side of the associativity axiom:

$$\frac{d}{dt} a_X(g_2 g_1, \theta) = D_g a_X\big|_{I,\theta}(h_1 + h_2).$$

Both sides are equal in the associativity axiom. Therefore, their derivatives are equal. That is,

$$D_\theta a_X\big|_{I,\theta}\Big(D_g a_X\big|_{I,\theta}(h_1)\Big) + D_g a_X\big|_{I,\theta}(h_2) = D_g a_X\big|_{I,\theta}(h_1 + h_2).$$

□

Theorem 6.3.2 captures how the composition of two infinitesimal actions generated by $h_1, h_2 \in \mathfrak{g}$ translates to a single infinitesimal action generated by their sum. When the group action is linear, $D_\theta a_X$ reduces to identity, and the infinitesimal action preserves the additive structure of the Lie algebra. When the group action is not linear, $D_\theta a_X$ introduces a correction term that modifies how the infinitesimal actions combine.

6.3.2 Examples

Below are examples of parameter space symmetry and their infinitesimal formulations. To facilitates calculation and interpretation, we express them in coordinate form. Assuming that $\mathsf{Param} = \mathbb{R}^n$, then for a single data point ($d = 1$), we can write equation 6.1 in coordinates as

$$\sum_{i=1}^{n} \sum_{k=1}^{dim(\mathfrak{g})} \frac{\partial L}{\partial \theta_i} \left(D_g a_X \big|_{I,\theta} \right)_{ik} h_k = 0. \tag{6.2}$$

Linear action of matrix groups

When $\mathsf{Param} = \mathbb{R}^n$ and G is a subgroup of $\mathrm{GL}(n)$ with a linear, data-independent symmetry $a_x(g,\theta) = g\theta$ for all $x \in X$, this coordinate form of loss invariance reduces to the equation in Theorem 3.1 in [104]. With $(D_g a)_{ijk} = \frac{\partial a_i}{\partial g_{jk}} = \delta_{ij}\theta_k$, Equation equation 6.2 becomes

$$\sum_{i=1}^{n} \sum_{k=1}^{n} \frac{\partial L}{\partial \theta_i} \theta_k h_{ik} = 0.$$

Our symmetry acts on parameters instead of data, but otherwise this matches Theorem 3.1 in [104].

Homogeneous two-layer neural network

We consider a homogeneous two-layer neural network with scalar weights for simplicity. Let parameter space $\mathsf{Param} = \mathbb{R}^2$ and data space $X \in \mathbb{R}$. Consider the loss function

$$L\colon \mathsf{Param} \times X \to \mathbb{R}, (w_1, w_2), x \mapsto w_2 \sigma(w_1 x)$$

with a homogeneous activation function $\sigma\colon \mathbb{R} \to \mathbb{R}$, i.e. $\sigma(\alpha x) = \alpha^c x$ for all $\alpha \in \mathbb{R}_{>0}$ and $x \in \mathbb{R}$, for some $c > 0$.

Let $G = (\mathbb{R}^\times, \times)$, and $\rho\colon G \to \mathrm{GL}_2, \alpha \mapsto \begin{pmatrix} \alpha & 0 \\ 0 & \alpha^{-c} \end{pmatrix}$. Then $a\colon \mathrm{GL}(2) \times \mathbb{R}^2 \to \mathbb{R}^2$,

$$\left(\rho(g), \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}\right) \mapsto \rho(g) \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \text{ is a symmetry of } L.$$

6.4 Building New Symmetry from Known Ones

One way to identify symmetries in a large network is by examining its components or subnetworks. Despite often having billions of parameters, neural networks typically consist of a limited set of function families, such as fully connected layers, attention mechanisms, and activation functions. This modular view suggests a mechanism by which symmetries in networks with fewer layers might extend to those in deeper networks. Additionally, within similar types of networks, it may be possible to extrapolate symmetries found in narrower layers to wider ones.

By focusing on symmetries in small architectures and using them to infer symmetries in larger ones, we circumvent the complexity associated with direct handling of high-dimensional parameter spaces. This approach not only simplifies the discovery of symmetries in large-scale networks but also provides a systematic method for using symmetries in smaller subnetworks to understand those in more extensive architectures. The subnetwork perspective has appeared in prevoius studies of the critical points in multilayer perceptrons [49, 128]. Proposition 6.4.1 generalizes this observation to subnetworks in arbitrary architectures and formalizes how to obtain symmetries of large networks from the symmetries of small ones.

If a loss function L depends on a subset of the parameters solely through a subnetwork f, then any symmetry of f will also preserve L:

Proposition 6.4.1. *Let* $L\colon \mathsf{Param} \times \mathscr{D}^d \to \mathbb{R}^d$ *where the parameter space* Param *is a product space* $\mathsf{Param} = \mathsf{Param}_1 \times \mathsf{Param}_2$. *Suppose for some spaces* S *and* T*, there exist functions* $h\colon \mathsf{Param}_1 \times \mathscr{D}^d \to S$, $f\colon \Theta_2 \times S \to T$ *and* $j\colon (\Theta_1 \times T) \times \mathscr{D}^d \to \mathbb{R}^d$*, such that for every* $\theta = (\theta_1, \theta_2) \in \mathsf{Param}$ *and* $X \in \mathscr{D}^d$, $L(\theta, X) = j\big((\theta_1, f\big(\theta_2, h(\theta_1, X)\big)), X\big)$. *If* $a\colon S \to (G \times \mathsf{Param}_2 \to \mathsf{Param}_2)$ *is a* G*-symmetry of* f*, then there is an induced* G*-symmetry of* L*,* $a'\colon \mathscr{D}^d \to (G \times \mathsf{Param} \to \mathsf{Param})$*, defined by* $a'_X(g, (\theta_1, \theta_2)) = \big(\theta_1, a_{h(\theta_1, X)}(g, \theta_2)\big)$.

Proof. We need to show that a' satisfies the identity and associative law of a group action and preserves L.

Since a is a group action on Param_2, it satisfies the identity axiom $a_{h(\theta_1,X)}(I,\theta_2) = \theta_2$. Applying this in the definition of a', we get $a'_X(I,(\theta_1,\theta_2)) = (\theta_1, a_{h(\theta_1,X)}(I,\theta_2)) = (\theta_1,\theta_2)$.

Since a is a group action on Param_2, it satisfies the associative law $a_{h(\theta_1,X)}(g_2g_1,\theta_2) = a_{h(\theta_1,X)}(g_2, a_{h(\theta_1,X)}(g_1,\theta_2))$, for all $g_1, g_2 \in G$. It follows that a' also satisfies the associative law. That is, $a'_X(g_2g_1,(\theta_1,\theta_2)) = (\theta_1, a_{h(\theta_1,X)}(g_2g_1,\theta_2)) = (\theta_1, a_{h(\theta_1,X)}(g_2, a_{h(\theta_1,X)}(g_1,\theta_2))) = a'_X(g_2, a'_X(g_1,(\theta_1,\theta_2)))$.

Finally, since a is a symmetry of f, we have $f(a_{h(\theta_1,X)}(g,\theta_2), h(\theta_1,X)) = f(\theta_2, h(\theta_1,X))$, for all $g \in G$. It follows that a' preserves the value of L: $L(a'_X(g,\theta),X) = j\big((\theta_1, f\big(a_{h(\theta_1,X)}(g,\theta_2), h(\theta_1,X)\big)\big),X\big) = j\big((\theta_1, f\big(\theta_2, h(\theta_1,X)\big)\big),X\big) = L(\theta,X)$. □

The relationship between the functions in the proposition is described by the commutative diagram below, where $p_1 \colon \Theta \to \Theta_1$, $p_2 \colon \Theta \to \Theta_2$ are projections onto Θ_1 and Θ_2, $\mathrm{id}_1 \colon \Theta_1 \to \Theta_1$ and $\mathrm{id}_2 \colon \Theta_2 \to \Theta_2$ are identity maps, and $X \in \mathscr{D}^d$ represents a batch of data. Space S and T can be interpreted as intermediate feature spaces in the neural network. When L can be decomposed in this way, the function h does not depend on Θ_2, and the function j depends on Θ_2 only through the output of f. This effectively confines L's dependency on Θ_2 to the transformation defined by f, ensuring that any transformation on Θ_2 not altering the output of f will not affect the output of L. Consequently, symmetries identified in the smaller network f can be extrapolated to the larger network L.

$$\begin{array}{ccccc} \Theta & & \xrightarrow{\qquad\qquad L(\cdot,X) \qquad\qquad} & & \mathbb{R}^d \\ \Big\downarrow{\scriptstyle p_1 \times p_2 \times p_1} & & & & \Big\uparrow{\scriptstyle j(\cdot,X)} \\ \Theta_1 \times \Theta_2 \times \Theta_1 & \xrightarrow{\mathrm{id}_1 \times \mathrm{id}_2 \times h(\cdot,X)} & \Theta_1 \times \Theta_2 \times S & \xrightarrow{\mathrm{id}_1 \times f(\cdot,\cdot)} & \Theta_1 \times T \end{array}$$

We apply Proposition 6.4.1 to construct symmetries in larger networks from those in smaller ones in the next two corollaries. Specifically, we show that some symmetries are

preserved as networks scale up through increasing the dimensionality of a layer or adding additional layers.

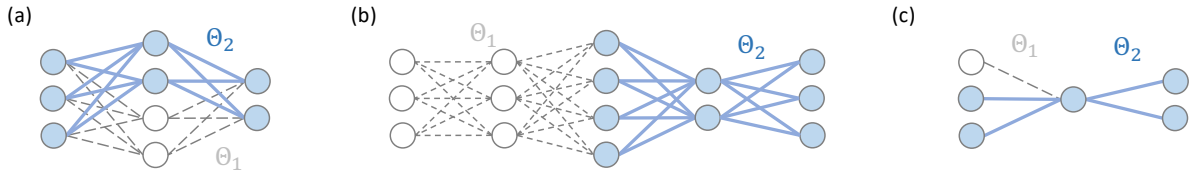


Figure 6.2. Examples of (a-b) when symmetries in subnetworks are symmetries in the original network, and (c) when they are not.

The first corollary describes how symmetries identified in narrower networks also apply to wider networks. A function $\sigma\colon \mathbb{R}^{h\times k} \to \mathbb{R}^{h\times k}$ is row-wise if, for any matrix $A \in \mathbb{R}^{h\times k}$ with rows $\{a_i \in \mathbb{R}^k\}_{i=1}^h$, the output matrix $\sigma(A)$ has rows $\{\sigma_{row}(a_i) \in \mathbb{R}^k\}_{i=1}^h$, where $\sigma_{row}\colon \mathbb{R}^k \to \mathbb{R}^k$ applies independently on each row of A. Element-wise functions are a special case of row-wise functions. For fully connected networks with row-wise activation functions, identifying a symmetry in one architecture suggests that the same symmetry will apply to wider versions of that architecture.

Corollary 6.4.2. *Consider a network parameter space* $\mathsf{Param}(m,h,n) = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$ *and data space* $\mathscr{D}(n,k) = \mathbb{R}^{n\times k}$. *Let* $\sigma\colon \mathbb{R}^{h\times k} \to \mathbb{R}^{h\times k}$ *be a row-wise function. Consider a function* $L_{mnhk}\colon \mathsf{Param}(m,h,n) \times \mathscr{D}(n,k) \to \mathbb{R}^{m\times k}$, *defined as* $L_{mnhk}((U,V),X) = U\sigma(VX)$ *for* $U \in \mathbb{R}^{m\times h}$, $V \in \mathbb{R}^{h\times n}$, *and* $X \in \mathbb{R}^{n\times k}$. *If there is a G-symmetry of* L_{mnhk}, *then there is a G-symmetry of* $L_{mnh'k}$ *with any* $h' > h$.

The next corollary shows that symmetries of a subset of layers are also symmetries in the entire network.

Corollary 6.4.3. *Let* $\mathsf{Param} = \mathsf{Param}_1 \times \ldots \times \mathsf{Param}_l$ *be a parameter space. Consider a list of spaces* $V_0 = \mathscr{D}^d$, $V_l = \mathbb{R}^d$, *and* V_1, ..., V_{l-1}. *Let* $L\colon \mathsf{Param} \times \mathscr{D}^d \to \mathbb{R}^d$ *be a function defined recursively by* $\{L_i\}_{i=1}^l$ *with* $L_i\colon \Theta_i \times V_{i-1} \to V_i$, *such that* $L = \phi_l$ *where* $\phi_i = L_i(\theta_i, \phi_{i-1}) \in V_i$ *and* $\phi_0 = X$. *If for some* $1 \le i \le l$, L_i *has a G-symmetry, then L has a G-symmetry.*

Both corollaries can be proved by factoring the parameter space and defining corresponding functions that compose to L, before applying Proposition 6.4.1. The explicit forms of h, f, and j are deferred to Appendix D. Figure 6.2 (a-b) shows the subset of parameters (Param_2) that the symmetry applies to in the corollaries. These are the subnetworks where symmetries are assumed to be known and which the larger network inherits. We also provide an example where symmetries defined on a subset of parameters may not be symmetries of the larger network (Figure 6.2 (c)). In this example, the output for the original networks requires information about Θ_2 in addition to the output of the subnetwork Θ_2 defines.

Note that this approach does not explore the emergence of new, more complex symmetries that may arise as the neural network scale up in size. Notably, there are cases where there exists a G symmetry over its input space, but group actions on individual layers are not loss-invariant ([84]). Nevertheless, studying smaller and simpler networks remains a effective strategy to obtain a significant number of symmetries in larger networks, and is a first step in characterizing the complete set of symmetries in modern architectures.

In addition to obtaining symmetries from those in smaller networks, we can also get symmetries for a loss function over data batches with a certain size, if we know there is a symmetry for this function over larger data batches. Concretely, if there exists a group action that preserves loss for all data batches of size $d \in \mathbb{Z}^+$, then that group action preserves loss for all data batches of size $d' < d$.

Proposition 6.4.4. *Let $L_d \colon \mathsf{Param} \times \mathscr{D}^d \to \mathbb{R}^d$ be a function that is applied pointwise on each of d data points in a data batch. If L_d admits a G-symmetry, then $L_{d'}$ admits a G-symmetry for all $d' < d$.*

Proof. Suppose that L_d has a G-symmetry. Let $a \colon \mathscr{D}^d \to (G \times \mathsf{Param} \to \mathsf{Param}), X_d \mapsto (a_{X_d} \colon g, \theta \mapsto \theta')$ be the corresponding group action. Define $a' \colon \mathscr{D}^{d'} \to (G \times \mathsf{Param} \to \mathsf{Param})$ by $X_{d'} \mapsto (a_{t(X_{d'})} \colon g, \theta \mapsto \theta')$, where $t \colon \mathscr{D}^{d'} \to \mathscr{D}^d$ appends $d - d'$ random data points to its input. Clearly, a' satisfies the identity and associate axiom and preserves loss. Therefore, a' is a

G-symmetry of $L_{d'}$. □

6.5 Automatic Discovery of Parameter Space Symmetry

Using the infinitesimal symmetry derived in the previous section, we construct an automated framework for discovering parameter space symmetries. Formulating symmetries in the infinitesimal form makes them easier to learn using an automatic framework, as it defines a set of local conditions for a function to be a symmetry. We then use this framework to find symmetries in several common architectures, and demonstrate the feasibility of finding symmetries in larger neural networks by examining their subnetworks.

6.5.1 Enforcing Loss Invariance and Group Axioms

Given a function L, our goal is to find a symmetry a and a set of Lie algebra elements h corresponding to a symmetry group of L. We parameterize a using a neural network with learnable parameters, and set h to be learnable as well. We define the following loss terms that quantify the deviation from loss invariance and the group axioms (identity and associativity law):

$$\begin{aligned}
\mathscr{L}_{\text{invariance}} &= \mathbb{E}_{x,\theta} |D_\theta L|_{\theta,X} \circ D_g a_X|_{I,\theta}(h)| \\
\mathscr{L}_{\text{id}} &= \mathbb{E}_{x,\theta} \|a_x(I,\theta) - \theta\|_2 \\
\mathscr{L}_{\text{assoc}} &= D_\theta a_X\big|_{I,\theta}\big(D_g a_X\big|_{I,\theta}(h_1)\big) \\
&\qquad + D_g a_X\big|_{I,\theta}(h_2) - D_g a_X\big|_{I,\theta}(h_1+h_2)
\end{aligned}$$

The three loss terms bias the action towards being loss-invariant, preserving identity, and satisfying the associativity property. By minimizing $\mathscr{L}_{\text{Lie\_deriv}}$, we ensure that the learned symmetry a and the Lie algebra element h satisfy the infinitesimal symmetry condition (Theorem 6.3.1). Minimizing $\mathscr{L}_{\text{id}}$ enforces the identity axiom, ensuring that the action of the identity element leaves the parameters unchanged. Minimizing $\mathscr{L}_{\text{assoc}}$ enforces the associative axiom (derivation in Appendix D).

By focusing on the Lie algebras, we enforce the loss invariance and group structure at the infinitesimal level. This formulation allows us to avoid computing exponential maps.

6.5.2 Regularizations

To prevent the learned group action from becoming trivial, we encourage the infinitesimal action to be nonzero. On the other hand, we do not want it to grow infinitely large for training stability. Therefore, in implementation, we include the following regularization term to encourage the norm of the infinitesimal action to be around a fixed positive real number β:

$$\mathscr{L}_{\text{reg\_id}} = \min_{a,h} \mathbb{E}_\theta |\beta - \|D_g a_X|_{I,\theta}(h)\||.$$

When learning multiple generators simultaneously, we want them to be orthogonal. Following [152], we do this by including the following cosine similarity between each pair of the k generators in the loss function:

$$\mathscr{L}_{\text{reg\_h\_orth}} = \sum_{1 \le i < j \le k} \frac{h_i \cdot h_j}{\|h_i\| \|h_j\|}.$$

Finally, we encourage sparsity of h for easier interpretation, with the regularization term

$$\mathscr{L}_{\text{reg\_h\_sparse}} = \sum_{k,j} |h_{kj}|.$$

The final training objective is a weighted average of symmetry loss and regularizations, with hyperparameters $\gamma_1, ..., \gamma_6 \in \mathbb{R}^+$:

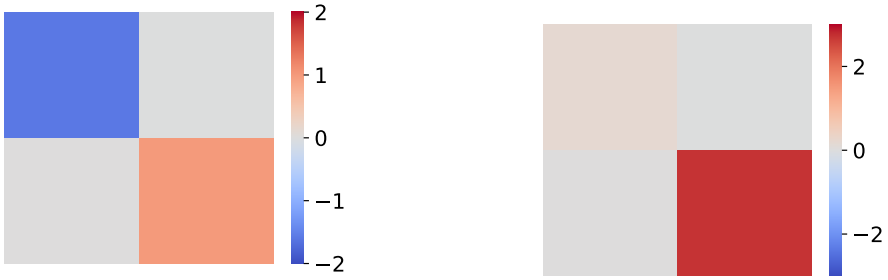
$$\begin{aligned} \min_{h,a} (&\gamma_1 \mathscr{L}_{\text{invariance}} + \gamma_2 \mathscr{L}_{\text{id}} + \gamma_3 \mathscr{L}_{\text{assoc}} + \gamma_4 \mathscr{L}_{\text{reg\_id}} \\ &+ \gamma_5 \mathscr{L}_{\text{reg\_h\_orth}} + \gamma_6 \mathscr{L}_{\text{reg\_h\_sparse}}). \end{aligned} \tag{6.3}$$

6.5.3 Learned Data-Independent Symmetries

We first validate that our method can learn generators for known data-independent symmetries in neural networks. We consider two-layer networks in the form of $L(W_1, W_2, X, Y) = \|W_2\sigma(W_1X) - Y\|^2$, where $W_2 \in \mathbb{R}^{m\times h}, W_1 \in \mathbb{R}^{h\times n}$ are parameters, $X \in \mathbb{R}^{n\times k}$, $Y \in \mathbb{R}^{m\times k}$ are data, and σ is a homogeneous activation function.

During training, we train the generators h and the group action a under objective equation 6.3. We parametrize a using a 4-layer MLP with hidden dimensions 128, 128, 128. The group aciton a takes a group element, parameter, and data as input and outputs transformed parameters. We use 2000 training samples, each containing a randomly generated set of parameters and data. We set the learning rate as 10^{-3} and the weights for the multi-objective loss as $\gamma_1 = 10$, $\gamma_2 = \gamma_4 = \gamma_5 = 1$, and $\gamma_6 = 0.1$.

As a proof of concept, we train for a group action and a single generator $h \in \mathbb{R}^{2\times 2}$ for the two-layer architecture with $m = h = n = k = 1$ and σ being the identity function. Figure 6.3(a) presents the learned generator h as a 2×2 matrix, which matches the expected generator that generates the rescaling group. Figure 6.3(b) visualizes a group element generated by h, corresponding to a rescaling group element.



(a) Generator h. (b) Group element $exp(h)$.

Figure 6.3. Learned generator for a two-layer linear MLP with scalar parameters and data, and a group element obtained via the exponential map.

Note that, however, we do not impose constraints on the group action (in particular, not enforcing linear actions). Hence we do not expect the learned generators to look similar

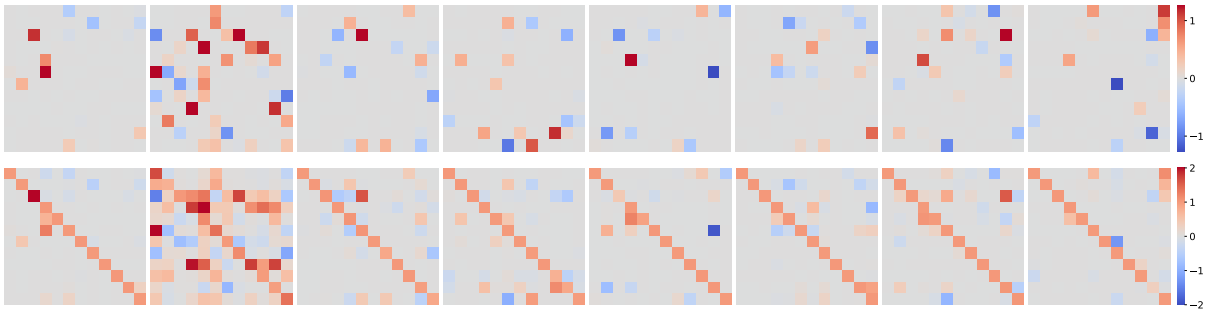


Figure 6.4. Learned data-dependent symmetries in a two-layer sigmoid MLP with parameters dimensions $W_2 \in \mathbb{R}^{3\times1}, W_1 \in \mathbb{R}^{3\times3}$ and data $X \in \mathbb{R}^{1\times3}, Y \in \mathbb{R}^{1\times1}$. Top: learned generators. Bottom: group elements obtained via the exponential map.

to the elements of the Lie algebra infinitesimal generators of the symmetry group in general. For example, the action a can be a composition of two function, the first transforming learned generators to the set of actual generators, and the second performing the group action.

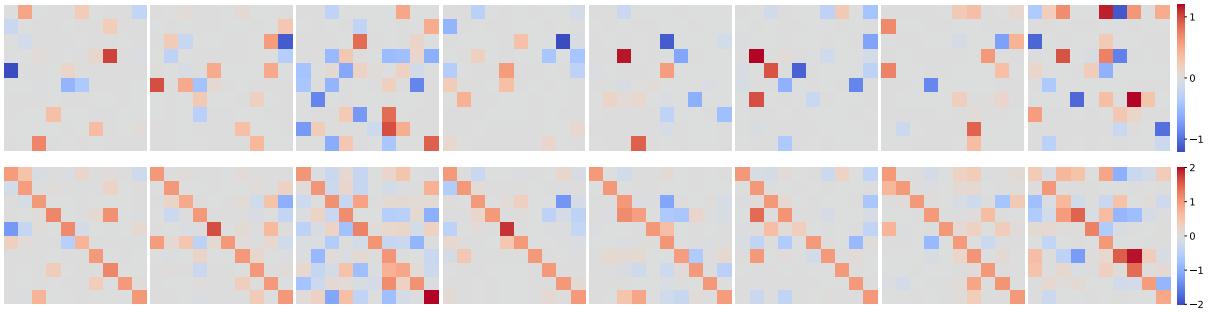


Figure 6.5. Learned data-dependent symmetries in a three-layer tanh MLP with parameters dimensions $W_1 \in \mathbb{R}^{2\times2}, W_2 \in \mathbb{R}^{2\times2}, W_3 \in \mathbb{R}^{2\times1}$ and data $X \in \mathbb{R}^{1\times2}, Y \in \mathbb{R}^{1\times1}$. Top: learned generators. Bottom: group elements obtained via the exponential map.

6.5.4 Learned Data-Dependent Symmetries

As a more practical application of our framework, we attempt to uncover data-dependent symmetries from architectures where no continuous symmetry is known before. We apply our framework to learn generators and loss-invariant group actions for two-layer neural network with sigmoid and tanh activation function, as well as a three-layer neural network with skip connection.

Specifically, we aim to learn symmetries in the two-layer networks defined in the previous section, but replacing σ by sigmoid or tanh. Our objective is again to find a set of generators h and a group action a that minimizes equation 6.3. We use 10000 training samples, each containing a randomly generated set of parameters and data. We set the learning rate as 10^{-3} and the weights for the multi-objective loss as $\gamma_1 = 1$, $\gamma_2 = \gamma_4 = 10$, $\gamma_5 = 1$, and $\gamma_6 = 0.1$.

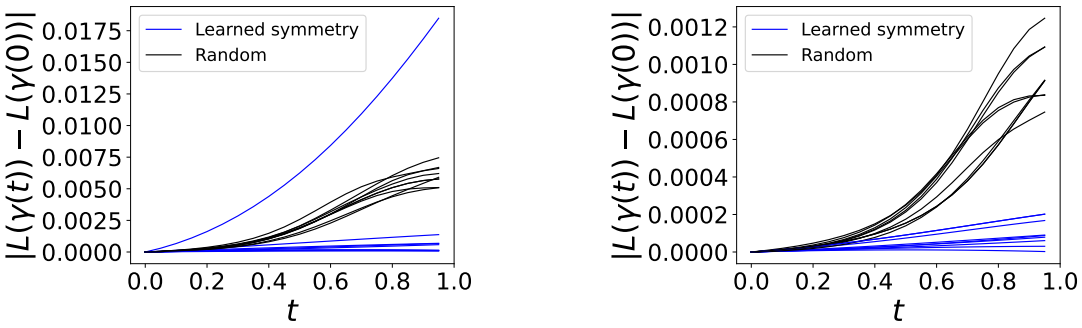
Figure 6.4 shows the learned generators for data-dependent symmetries in a two-layer sigmoid MLP with parameters dimensions $W_1 \in \mathbb{R}^{3\times3}, W_2 \in \mathbb{R}^{3\times1}$ and data $X \in \mathbb{R}^{3\times1}, Y \in \mathbb{R}^{1\times1}$. Sigmoid networks have no data-independent continuous symmetry, and we observed that the derivative of the learned group action with respect to data is indeed nonzero. Therefore, this set of symmetries are data-dependent, indicating that our method successfully learns data-dependent symmetries for this architecture.

Figure 6.5 shows the learned generators for data-dependent symmetries in a three-layer tanh MLP with parameters dimensions $W_1 \in \mathbb{R}^{2\times2}, W_2 \in \mathbb{R}^{2\times2}, W_3 \in \mathbb{R}^{2\times1}$ and data $X \in \mathbb{R}^{1\times2}, Y \in \mathbb{R}^{1\times1}$. The generators indicate the existence of symmetries that act on non-contiguous layers, which has not been discussed in previous literature. These discoveries may motivate future mathematical work on the structure of neural network symmetry groups.

Although the loss invariance is only enforced locally during learning, the learned generators approximately preserves loss over the group actions using group elements from these generators. To verify loss invariance, we plot the loss on curves generated from the learned symmetries (Figure 6.6). Specifically, we compute the loss value on curves $\gamma(t) = a_X(\exp(ht), \theta)$ in Param, where h is a learned Lie algebra element and a_X is the learned group action. Compared to the loss on curves generated by random Lie algebra elements, the loss variation along curves generated by the learned symmetries is consistently lower.

6.5.5 Symmetry in Transformers

We study a minimal transformer $f_\theta : \mathbb{R}^{B\times T\times d} \to \mathbb{R}^{B\times T\times d}$ with batch size $B = 8$, sequence length $T = 2$, and model dimension $d = 2$. The model consists of a single self-attention block



(a) 2-layer sigmoid MLP.

(b) 3-layer tanh MLP.

Figure 6.6. Loss on curves generated by learned symmetries vs. loss on random curves in the parameter space.

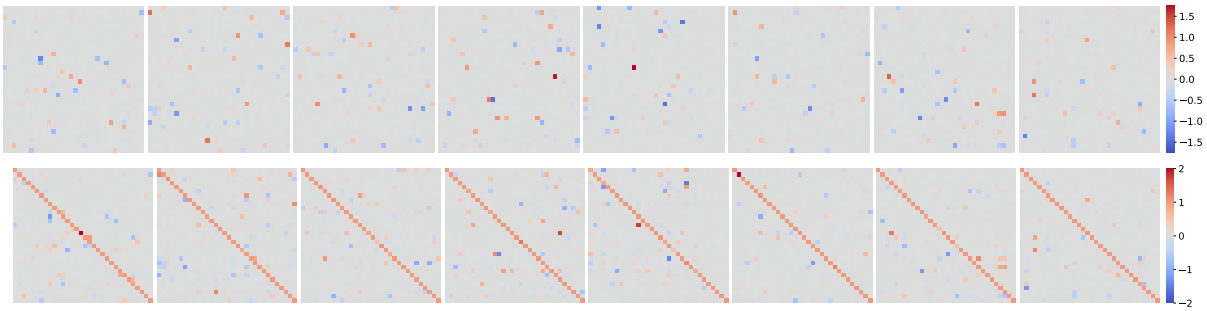


Figure 6.7. Learned data-dependent symmetries in a tiny transformer. Top: learned generators. Bottom: group elements obtained via the exponential map.

with one attention head and a position-wise feedforward network of hidden dimension $d_{\text{ff}} = 4$. Given input $X \in \mathbb{R}^{B\times T\times d}$, self-attention is computed as

$$\text{Attn}(X) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V,$$

where $Q = XW_Q, K = XW_K, V = XW_V$, and $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$, followed by a residual connection and feedforward map

$$f_\theta(X) = \text{FFN}\big(X + \text{Attn}(X)\big), \quad \text{FFN}(Z) = ZW_2W_1,$$

with $W_1 \in \mathbb{R}^{d \times d_{\text{ff}}}$ and $W_2 \in \mathbb{R}^{d_{\text{ff}} \times d}$. The full parameter vector is $\theta = \{W_Q, W_K, W_V, W_1, W_2\} \in \mathbb{R}^p$ with $p = 32$. For each training step, parameters θ are freshly initialized, and data tensors $X, Y \in \mathbb{R}^{B \times T \times d}$ are sampled i.i.d. from a standard normal distribution. We learn a data-dependent parameter-space group action and $n_h = 8$ generators using an action MLP with three hidden layers of width 128, trained for 2000 steps with Adam at learning rate 10^{-3}.

Figure 6.7 visualizes the learned generators h_i and their corresponding group elements $exp(h_i)$, shown as heatmaps over the flattened parameter space of the transformer. The learned generators exhibit structured, nontrivial coupling patterns across parameters that are not explained by known data-independent symmetries of transformer architectures. This suggests the presence of previously unexplored data-dependent symmetries in transformer models, which our method is able to uncover. Figure 6.7 shows the generators and symmetry group elements found. These symmetries do not correspond to known ones in transformers and suggest that there exists novel data-dependent symmetries to exploit.

6.6 Discussion

In this work, we introduce a framework for formalizing and discovering parameter space symmetries in neural networks. By deriving infinitesimal forms of these symmetries, we enable automated identification of loss-invariant transformations, including previously unknown ones. Additionally, we generalizes symmetries from smaller subnetworks to larger architectures, reducing computational complexity while improving scalability. These findings provide a foundational step toward a deeper understanding of data-dependent parameter space symmetries, uncovering new structural insights that could reshape how we analyze and optimize neural networks.

While our results suggest that there are previously unknown data-dependent symmetries in various neural network architectures, the existence and number of symmetries in neural network parameter spaces remain open questions. Whether the number of symmetries is affected

by existence of symmetry in data or changes during training are also interesting directions. Future work will examine the structure of learned symmetry, such as the dimension of Lie algebras.

The discovered symmetries could be useful in downstream applications such as weight space learning. When the parameters of a neural network are used as input to make predictions, parameter space symmetry of the input network becomes the data space symmetry of processing network. Architectures that can respect the symmetries in the input networks has proven to be able to process neural network weights effectively [156, 93, 74, 137]. Leveraging the learned symmetries via equivariant architectures [45] or frame averaging [117] can potentially improve current approaches that are limited to known symmetries.

Acknowledgments

This chapter, in part, is based on the paper from Zhao, Bo; Dehmamy, Nima; Walters, Robin; Yu, Rose. “Finding Symmetry in Neural Network Parameter Spaces.” currently being prepared for submission. The dissertation author is the primary investigator and author of this paper.

Chapter 7

Discussion and Future Directions

Symmetry is prevalent in neural network parameter spaces and appears in many areas of machine learning, though often overlooked. Recognizing and formalizing these symmetries connects deep learning to well-established mathematical tools from group theory and geometry. Symmetry's intrinsic connection to structure improves our understanding of how neural networks work and hints at better architectures, faster optimization techniques, and new approaches to solving problems with real-world impact.

Symmetry is, of course, not the only path forward in the study of neural networks. Like modeling training dynamics using classical mechanics or designing new neural networks with inspirations from neuroscience, it is an example of approaching problems in machine learning from the view of a pre-existing subject, allowing us to bring existing tool and insight to bear on the problem. What makes a mathematical approach especially appealing is that neural networks are fully artificial, abstract objects, unconstrained by the specific laws of physics or biology. By examining parameter space symmetry, we gain access to powerful mathematical frameworks that help us better understand these systems and ultimately design more effective models.

In this section, we outline key research directions towards a more complete theoretical foundation of parameter space symmetry, better understanding of the role of symmetry in deep learning theory, and broader applicability of symmetry-informed methods.

7.1 Mathematical Foundations

A fundamental challenge in understanding parameter space symmetry is to develop a complete and rigorous characterization of all symmetries that preserve neural-network functions. From a mathematical perspective, clarifying the existence, completeness, and structures of these symmetry groups represents a foundational problem. Even seemingly simple network architectures can have nontrivial symmetry groups, and identifying them is essential for understanding loss landscapes and optimization dynamics.

Another challenge is extending the notion of symmetry from exact, function-preserving transformations to more flexible, data-dependent symmetries. In many neural networks with elementwise activation functions, all transformations that do not change the function are known. However, this set of symmetry is often small, as they are required to keep the loss invariant for all input values. Data-dependent symmetry allows for larger symmetry groups, but current analysis is limited to symmetry that preserves the loss for a single data point. Investigating the existence and structure of data-dependent symmetry for different batch sizes will to make this set of symmetry more relevant to practical settings.

Thus far, most work has focused on layer-wise equivariances arising from adjacent activations, potentially overlooking more global invariances. Existing studies examine small components—pairs of layers whose activations admit a group action—but this does not preclude broader classes of symmetry that act across multiple, non-adjacent layers or repeated architectural blocks. Exploring such symmetries could substantially expand the known symmetry groups and reveal new geometric or topological structures in parameter space.

Besides pursuing a general theory, detailed investigations into architecture-specific symmetries are also valuable. Models such as neural radiance fields [100], which are computationally expensive to optimize, might benefit greatly from symmetry-informed optimization methods. Discovering and characterizing symmetries particular to these architectures would provide concrete opportunities to accelerate training and enhance scalability, thereby translating theoretical

advances into immediate practical improvements.

Finally, numerical and visualization tools can help guide theoretical characterization of parameter space symmetries. Methods for numerically constructing and visualizing functionally equivalent parameters, such as those developed by [91], provide practical insight into the connectedness, dimensionality, and topological complexity of symmetry-induced loss level sets. Numerical discovery of symmetries [158] and symmetry-induced structures [97] in the parameter space may also provide intuitions on the existence and number of symmetries in a given architecture. These computational approaches can be important in guiding theoretical efforts in large-scale architectures.

7.2 Deep Learning Theory

Parameter space symmetry is not merely a mathematical curiosity—it offers a lens to understand core phenomena in deep learning theory. As neural networks scale in size and complexity, symmetry provides a principled foundation for analyzing learning dynamics, the geometry of loss landscapes, and model capacity. Symmetry considerations are thus increasingly central in explaining how overparameterized networks behave, generalize, and learn in practice.

Training dynamics and implicit bias. One promising direction is to study learning dynamics through the lens of symmetry and conserved quantities. As we have seen in Section 2, continuous symmetries induce conserved quantities that remain fixed along gradient flows and partially parameterize the minimum. These quantities label the optimizer's position and reveal how initialization and symmetry constrain the final solution, formalizing implicit bias. However, the precise relationship between conserved quantities, implicit bias, and generalization remains poorly understood—especially in deeper architectures or when SGD breaks these symmetries due to finite-step updates. Understanding how stochasticity or regularization cause the drift of these conserved quantities will be an alternative route to explain phenomena in training neural networks, such as why stochastic gradient descent prefers minima that generalize.

Loss landscape geometry and generalization. Symmetry is an integral part of the geometry of the loss landscape, which is closely linked to generalization. By expanding minima into manifolds of equivalent solutions, symmetries create flat directions that confound curvature-based metrics such as sharpness. Recent work by [32] addresses this issue by formulating sharpness on a Riemannian quotient manifold that factors out the continuous symmetry of transformers, recovering a strong correlation between curvature and generalization. Developing such symmetry-aware geometric formalisms may help reconcile conflicting observations about the relationship between flatness and generalization.

Expressivity and approximation theory. Symmetry has direct implications for the expressivity and effective capacity of neural networks. Since many parameter configurations can represent the same function (e.g., due to permutation symmetry in multilayer perceptrons), the function class is smaller than suggested by raw parameter counts. Recent work by [126] quantifies this, showing that factoring out symmetry leads to significantly tighter covering number bounds—improving estimates by factorial factors in width. These results suggest that traditional complexity measures overestimate model capacity by ignoring symmetric redundancies. Extending such analyses to broader symmetry groups (beyond permutations) could yield sharper generalization bounds by treating symmetry-related parameters as equivalent in the function space.

Introducing or removing parameter symmetries? Architectural design offers a means to either introduce or eliminate symmetries, which provides the opportunity to examine symmetry's effect on various aspect of deep learning. On one hand, imposing symmetry can enhance training stability and invariance. For example, [92] introduce a scale-invariant transformer, replacing Softmax with a normalized ReLU to create a model trainable by plain SGD yet competitive with Adam-trained BERT variants. On the other hand, removing symmetry reduces degeneracies, creates more connected minima, and sometimes makes optimization easier. [94] propose architectures that break permutation symmetry between neurons, enabling linear mode connectivity and improving Bayesian training efficiency. Additionally, [170] show

that removing symmetries via small perturbations can prevent capacity collapse and improve both optimization and generalization by encouraging exploration of more expressive model configurations. An important open question is determining which symmetries to preserve and which to break, in order to best align model behavior with the goals of a given learning task.

Taken together, these directions position symmetry not only as a descriptive tool for neural networks, but as a foundational principle for understanding learning dynamics, designing architectures, and guiding future theory in deep learning.

7.3 Applications

Optimization on loss level sets. One practical use of symmetry is to facilitate movement along loss level sets in the parameter space. Points on a fiber of the realization map can have very different neighborhoods [58]. In other words, points on the same level set can be different. Early work illustrates the usefulness of this property in optimization [11]. Despite initial promise, these methods have not been wide adapted, possibly due to a lack of large-scale benchmark study and an easy to use implementation, as well as the mathematical background required to adapt these algorithms to new architectures. New implementation techniques [102, 148] and learning the teleportation destination [161, 154] might help reduce the cost of teleportation in larger scale applications.

Besides scaling up teleportation, another promising direction is to explore level sets via symmetry transformation in diverse domains. When some parts of the minima are better than others, one can use symmetry to explore the minima to find the better models. One example is continual learning, where one optimize in the minimum manifold [42] and could be a method for fine-tuning pre-trained models. Another example is model alignment, where symmetry is used to move models closer in the parameter space to improve model fusion [3, 155]. Recent work also demonstrates that parameter space symmetries can be leveraged to align sparse subnetworks across different random initializations, enabling effective sparse training from

scratch [2]. Moreover, some minima may produce lower error during quantization than others [98, 105, 86]. Given a known symmetry and a point on minimum, we can search for these better points on an orbit.

Reducing search space in sampling. Symmetry can be exploited beyond standard gradient-based training, for instance in Bayesian methods and sampling-based optimization [144]. The parameter space explored by sampling algorithms (such as MCMC for neural network weights) may also contain redundancies coming from symmetry, though this aspect remains less studied. Factoring out symmetries in such scenarios could reduce the effective search space and improve sampling efficiency. For example, if parameters related by a continuous symmetry produce the same likelihood, one could constrain the sampler to move only in the reduced space of unique configurations, thereby eliminating needless exploration of equivalent states. This idea can also be used as a diagnostic: known symmetries provide a consistency check for the quality of sampling (the sampler should visit all members of a symmetry orbit with equal probability). Discrete symmetry has been explored in depth [144, 88, 150], but factoring out continuous symmetry might be more appealing, since it reduces the dimension, instead of just volume, of the posterior space.

Acknowledgments

This chapter, in part, is based on the paper published as: Zhao, Bo; Walters, Robin; Yu, Rose. "Symmetry in Neural Network Parameter Spaces." Transactions on Machine Learning Research, issn 2835-8856 (2026). The dissertation author is the primary investigator and author of this paper.

Appendix A

Supplementary Material for Chapter 2

A.1 Relation to Noether's theorem

We will now show how the approach in [134] relates to our conservation law $dQ/dt = \left\langle \dot{\boldsymbol{\theta}}, M\boldsymbol{\theta} \right\rangle = 0$. Assuming a small time-step $\tau \ll 1$, we can write GD as $\boldsymbol{\theta}(t+\tau) - \boldsymbol{\theta}(t) = -\tilde{\varepsilon}\nabla L(\boldsymbol{\theta}(t))$. Expanding the l.h.s to second order in τ and discarding $O(\tau^3)$ terms defines the 2nd order GF equation

$$\text{2nd order GF:} \qquad \frac{d\boldsymbol{\theta}}{dt} + \frac{\tau}{2}\frac{d^2\boldsymbol{\theta}}{dt^2} = -\varepsilon\nabla L. \tag{A.1}$$

Here $\varepsilon = \tilde{\varepsilon}/\tau$. To use Noether's theorem, the dynamics (i.e. GF) must be a variational (Euler-Lagrange (EL)) equation derived from an "action" $S(\boldsymbol{\theta})$ (objective function), which for equation A.1 is the time integral of Bregman Lagrangian [143] L

$$S(\boldsymbol{\theta}) = \int dt L(\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t); t) = \int_{\gamma} dt e^{t/\tau}\left[\frac{\tau}{2}\left\langle \dot{\boldsymbol{\theta}}, \varepsilon^{-1}\dot{\boldsymbol{\theta}} \right\rangle - L(\boldsymbol{\theta})\right] \tag{A.2}$$

where $\boldsymbol{\theta} : \mathbb{R} \to \mathsf{Param}$ is a trajectory (flow path) in Param, parametrized by t. The variational EL equations find the paths γ^* which minimize the action, meaning $\partial S_\gamma/\partial\gamma|_{\gamma^*} = 0$.

Noether's theorem

states that if $M \in \mathfrak{g}$ is a symmetry of the *action* $S(\boldsymbol{\theta})$ equation A.2 (not just the loss $L(\boldsymbol{\theta})$), then the Noether current J_M is conserved

$$\begin{aligned} \text{Noether current:} \quad & J_M = \left\langle \overline{M}_{\boldsymbol{\theta}}, \frac{\partial L}{\partial \dot{\boldsymbol{\theta}}} \right\rangle = e^{t/\tau} \left\langle \overline{M}_{\boldsymbol{\theta}}, \varepsilon^{-1} \dot{\boldsymbol{\theta}} \right\rangle = e^{t/\tau} J_{0M}, \\ \text{Conservation:} \quad & \frac{dJ_M}{dt} = e^{t/\tau} \left[\frac{1}{\tau} J_{0M} + \frac{dJ_{0M}}{dt} \right] = 0, \quad \Rightarrow \quad J_{0M}(t) = J_{0M} e^{-t/\tau} \end{aligned} \tag{A.3}$$

[134] also derived the Noether current equation A.3, but concludes that because $L(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \neq L(\boldsymbol{\theta})$, the symmetries are "broken" and therefore doesn't derive conserved charges for the types of symmetries we discussed above. However, while [134] focuses on 2nd order GF, we note that our conserved Q were derived for *first order GF*, which is found from the $\tau \to 0$ limit of 2nd oder GF. In this limit $L \to e^{t/\tau} L$ and thus symmetries of L also becomes symmetries of L. When $\tau \to 0$, 2nd order GF reduces to $\varepsilon^{-1} \dot{\boldsymbol{\theta}} = -\nabla L$ the conserved charge goes to

$$\lim_{\tau \to 0} J_{0M} = \left\langle \overline{M}_{\boldsymbol{\theta}}, \nabla L \right\rangle = J_{0M}(0) \lim_{\tau \to 0} e^{-t/\tau} = 0, \tag{A.4}$$

which means that we recover the invariance under infinitesimal action equation 2.6. In fact, for linear symmetries and symmetric $M \in \mathfrak{g}$, $J_{0M} = dQ_M/dt = 0$.

A.2 Neural networks: linear group actions

In this appendix, we provide an extended discussion of the topics of Section 2.3, including full proofs of all results. Specifically, after some technical background material on Jacobians and differentials, we specify our conventions for neural network parameter space symmetries. In contrast to the discussion of the main text, we (1) assume that neural networks have biases, and (2) focus on the multi-layer case rather than just the two-layer case. We then turn our attention to group actions of the parameter space that leave the loss invariant, and the resulting infinitesimal symmetries. The groups we consider are all subgroups of a large group of change-of-basis

transformations of the hidden feature spaces; we call this group the 'hidden symmetry group'. We also compute the dimensions of generic extended flat minima in various relevant examples. Finally, we explore consequences of invariant group actions for conserved quantities.

A.2.1 Jacobians and differentials

In this section, we summarize background material on Jacobians and differentials. We adopt notation and conventions from differential geometry. Let $U \subset \mathbb{R}^n$ be an open subset of Euclidean space $\mathbb{R}^n$, and let $F : U \to \mathbb{R}^m$ be a differentiable function. Let $F_1, \ldots, F_m : U \to \mathbb{R}$ be the components of F, so that $F(u) = (F_1(u), \ldots, F_m(u))$. The Jacobian of F, also know as differential of F, at $u \in U$ is the following matrix of partial derivatives evaluated at u:

$$dF_u = \begin{bmatrix} \frac{\partial F_1}{\partial_{x_1}}\big|_u & \frac{\partial F_1}{\partial_{x_2}}\big|_u & \cdots & \frac{\partial F_1}{\partial_{x_n}}\big|_u \\ \frac{\partial F_2}{\partial_{x_1}}\big|_u & \frac{\partial F_2}{\partial_{x_2}}\big|_u & \cdots & \frac{\partial F_2}{\partial_{x_n}}\big|_u \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial_{x_1}}\big|_u & \frac{\partial F_m}{\partial_{x_2}}\big|_u & \cdots & \frac{\partial F_m}{\partial_{x_n}}\big|_u \end{bmatrix}$$

The differential dF_u defines a linear map from $\mathbb{R}^n$ to $\mathbb{R}^m$, that is, an element of $\mathbb{R}^{m\times n}$. Observe that if F itself is linear, then, as matrices, $dF_u = F$ for all points $u \in U$. If $G : \mathbb{R}^m \to \mathbb{R}^p$ is another differentiable map, then the chain rule implies that, for all $u \in \mathbb{R}^n$, we have:

$$d(G \circ F)_u = dG_{F(u)} \circ dF_u.$$

In the special case the $m = 1$, the differential is a $1 \times n$ row vector, and the *gradient* $\nabla_u F$ of F at $u \in \mathbb{R}^n$ is defined as the transpose of the Jacobian dF_u:

$$\nabla_v F = (dF_u)^T = \begin{bmatrix} \frac{\partial F}{\partial x_1}\big|_u & \cdots & \frac{\partial F}{\partial x_n}\big|_u \end{bmatrix}^T = \left(\frac{\partial F}{\partial x_i}\Big|_u \right)_{i=1}^n \in \mathbb{R}^n$$

A.2.2 Neural network parameter spaces

Consider a neural network with L layers, input dimension $n_0 = n$, output dimension $n_L = m$, and hidden dimensions given by $n_1, \ldots, n_{L-1}$. For convenience, we group the dimensions into a tuple $\mathbf{n} = (n_0, n_1, \ldots, n_L)$. The parameter space is given by:

$$\mathsf{Param}(\mathbf{n}) = \mathbb{R}^{n_L \times n_{L-1}} \times \mathbb{R}^{n_{L-1} \times n_{L-2}} \times \cdots \times \mathbb{R}^{n_2 \times n_1} \times \mathbb{R}^{n_1 \times n_0} \times \mathbb{R}^{n_L} \times \mathbb{R}^{n_{L-1}} \times \cdots \times \mathbb{R}^{n_1}.$$

We write an element therein as a pair $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ of tuples $\mathbf{W} = (W_L, \ldots, W_1)$ and $\mathbf{b} = (b_L, \ldots, b_1)$, so that W_i is a $n_i \times n_{i-1}$ matrix and b_i is a vector in $\mathbb{R}^{n_i}$ for $i = 1, \ldots, L$. When $\mathbf{n}$ is clear from context, we write simply Param for the parameter space. Fix a piecewise differentiable function $\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ for each $i = 1, \ldots, L$. The activations can be pointwise (as is conventionally the case), but are not necessarily so. The feedforward function

$$F = F_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^m$$

corresponding to parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}) \in \mathsf{Param}$ with activations σ_i is defined in the usual recursive way. To be explicit, we define the partial feedforward function $F_{\boldsymbol{\theta},i} = F_i : \mathbb{R}^n \to \mathbb{R}^{n_i}$ to be the map taking $x \in \mathbb{R}^n$ to $\sigma_i(W_i F_{i-1}(x) + b_i)$, for $i = 1, \ldots, L$, with $F_0 = \mathrm{id}_{\mathbb{R}^{n_0}}$. Then the feedforward function is $F = F_L$. The "loss function" L of our model is defined as:

$$L : \mathsf{Param} \times \mathbf{Data} \to \mathbb{R}, \qquad L(\boldsymbol{\theta}, (x, y)) = \mathrm{Cost}(y, F_{\boldsymbol{\theta}}(x)). \tag{A.5}$$

where $\mathbf{Data} = \mathbb{R}^{n_0} \times \mathbb{R}^{n_L}$ is the space of data (i.e., possible training data pairs), and $\mathrm{Cost} : \mathbb{R}^{n_L} \times \mathbb{R}^{n_L} \to \mathbb{R}$ is a differentiable cost function, such as mean square error or cross-entropy. Many, if not most, of our results involve properties of the loss function L that hold for any training data. Hence, unless specified otherwise, we take a fixed batch of training data $\{(x_i, y_i)\}_{i=1}^k \subseteq \mathbf{Data}$, and consider the loss to be a function of the parameters only.

The above constructions generalize to multiple samples. Specifically, instead of $x \in \mathbb{R}^{n_0}$ and $y \in \mathbb{R}^{n_L}$, one has matrices $X \in \mathbb{R}^{n_o \times k}$ and $Y \in \mathbb{R}^{n_L \times k}$ whose columns are the k samples. Additionally, one uses the Frobenius norm of $n_L \times k$ matrices to compute the loss function. The i-th partial feedforward function is $F_{\boldsymbol{\theta},i} : \mathbb{R}^{n_0 \times k} \to \mathbb{R}^{n_i \times k}$, and we have the feedforward function $F_{\boldsymbol{\theta}} : \mathbb{R}^{n_0 \times k} \to \mathbb{R}^{n_L \times k}$. (We use the same notation as in the case $k = 1$; the number of samples will be understood from context.)

Example A.2.1. *Consider the case $L = 2$ with no biases and no output activation. The dimension vector is $(n_0, n_1, n_2) = (n, h, m)$, so the parameter space is $\mathsf{Param}(n,h,m) = \mathbb{R}^{h \times n} \times \mathbb{R}^{m \times h}$. Taking the mean square error cost function, the loss function for data $(X,Y) \in \mathbb{R}^{n \times k} \times \mathbb{R}^{m \times k}$ takes the form $L(\boldsymbol{\theta}, (X,Y)) = \frac{1}{k}\|Y - U\sigma(VX)\|^2$, where $\boldsymbol{\theta} = (W_2, W_1) = (U,V) \in \mathsf{Param}$.*

A.2.3 Action of continuous groups and infinitesimal symmetries

Let G be a group. An action of G on the parameter space Param is a function $\cdot : G \times \mathsf{Param} \to \mathsf{Param}$, written as $g \cdot \boldsymbol{\theta}$, that satisfies the unit and multiplication axioms of the group, meaning $I \cdot \boldsymbol{\theta} = \boldsymbol{\theta}$ where I is the identity of G, and $g_1 \cdot (g_2 \cdot \boldsymbol{\theta}) = (g_1 g_2) \cdot \boldsymbol{\theta}$ for all $g_1, g_2 \in G$. Recall that we say an action $G \times \mathsf{Param} \to \mathsf{Param}$ is a *symmetry* of Param with respect to L if it leaves the loss function invariant, that is:

$$L(g \cdot \boldsymbol{\theta}) = L(\boldsymbol{\theta}), \qquad \forall \boldsymbol{\theta} \in \mathsf{Param}, \quad g \in G \tag{A.6}$$

The groups within the scope of this paper are all matrix Lie groups, which are topologically closed subgroups $G \subseteq \mathrm{GL}_n(\mathbb{R})$ of the general linear group of invertible $n \times n$ real matrices. Any smooth action of such a group induces an action of the infinitesimal generators of the group, i.e., elements of its Lie algebra. Concretely, let $\mathfrak{g} = \mathrm{Lie}(G) = T_I G$ be the Lie algebra, which can be thought of as a certain subspace of matrices in $\mathfrak{gl}_n = \mathbb{R}^{n \times n}$, or (equivalently) as the tangent space[1]. at the identity I of G. For every matrix $M \in \mathfrak{g}$, we have an exponential map $\exp_M : \mathbb{R} \to G$

[1]Hence, elements of the Lie algebra are 'velocities' at the identity of G. More precisely, for every Lie algebra element ξ, there is a path $\gamma_\xi : (-\varepsilon, \varepsilon) \to G$ whose value at 0 is the identity of G and whose derivative (i.e., velocity)

defined as $\exp_M(t) = \sum_{k=0}^{\infty} \frac{(tM)^k}{k!}$. Given an action of G on Param, the *infinitesimal action* of $M \in \mathfrak{g}$ is a vector field $\overline{M}$:

$$\text{Infinitesimal action of } M \text{ vector field:} \quad \overline{M}_{\boldsymbol{\theta}} := \left.\frac{d}{dt}\right|_{t\to 0} (\exp_M(t)\cdot\boldsymbol{\theta}), \qquad \forall \boldsymbol{\theta} \in \mathsf{Param}. \quad \text{(A.7)}$$

Hence, the value of the vector field $\overline{M}$ at the parameter value $\boldsymbol{\theta}$ is given by the derivative at zero of the function $t \mapsto (\exp_M(t)\cdot\boldsymbol{\theta})$. In the case of a parameter space symmetry, the invariance of L translates into the orthogonality condition in Proposition 2.3.9, where the inner product $\langle , \rangle : \mathsf{Param} \times \mathsf{Param} \to \mathbb{R}$ is calculated by contracting all indices, e.g. $\langle A, B\rangle = \sum_{ijk\ldots} A_{ijk\ldots} B_{ijk\ldots}$.

Proof of Proposition 2.3.9. The gradient is the transpose of the Jacobian (see Section A.2.1), so the left-hand-side becomes $dL_{\boldsymbol{\theta}}\left(\frac{d}{dt}\big|_0 (\exp_M(t)\cdot\boldsymbol{\theta})\right)$. We compute:

$$dL_{\boldsymbol{\theta}}\left(\left.\frac{d}{dt}\right|_0 (\exp_M(t)\cdot\boldsymbol{\theta})\right) = \left.\frac{d}{dt}\right|_0 L(\exp_M(t)\cdot\boldsymbol{\theta}) = \left.\frac{d}{dt}\right|_0 L(\boldsymbol{\theta}) = 0$$

where the first equality follows by the chain rule, the second equality uses the invariance of L, and the third equality follows since $L(\boldsymbol{\theta})$ does not depend on t. □

Next, we comment on the case of a linear action. Observe that the parameter space is a vector space of dimension $d = \dim(\mathsf{Param}) = \sum_{i=1}^{L} n_i(1+n_{i-1})$. Hence, there is an isomorphism $\mathsf{Param} \simeq \mathbb{R}^d$ which flattens any tuple $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ into a vector in $\mathbb{R}^d$. We can identify the group $\mathrm{GL}(\mathsf{Param})$ of all invertible linear transformations of the parameter space with the group $\mathrm{GL}_d(\mathbb{R})$ of invertible $d \times d$ matrices, and the Lie algebra of $\mathrm{GL}(\mathsf{Param})$ with $\mathfrak{gl}_d = \mathbb{R}^{d\times d}$. Suppose G acts linearly on the parameter space. Then we can identify[2] G with a subgroup of $\mathrm{GL}(\mathsf{Param})$, acting on Param with matrix multiplication. Similarly, we can identify the Lie algebra $\mathfrak{g}$ of G with a Lie subalgebra of $\mathfrak{gl}_d = \mathbb{R}^{d\times d}$. In this case, the infinitesimal action is given by matrix multiplication: $\overline{M}_{\boldsymbol{\theta}} = M \cdot \boldsymbol{\theta}$. We recover Equation equation 2.6.

at zero is ξ.

[2]Modding out by the kernel, if necessary.

A.2.4 The hidden symmetry group

Consider a neural network with L layers and dimensions $\mathbf{n}$. The *hidden symmetry group* corresponding to dimensions $\mathbf{n}$ is defined as the following product of general linear groups:

$$\mathrm{GL}_{\mathbf{n}^{\text{hidden}}} = \mathrm{GL}_{n_1} \times \cdots \times \mathrm{GL}_{n_{L-1}}$$

An element is a tuple of invertible matrices $\mathbf{g} = (g_1, \ldots, g_{L-1})$, where $g_i \in \mathrm{GL}_{n_i}$. Consider the action of the hidden symmetry group on the parameter space given by

$$\mathrm{GL}_{\mathbf{n}^{\text{hidden}}} \circlearrowright \mathsf{Param}(\mathbf{n}) \qquad \mathbf{g} \cdot (\mathbf{W}, \mathbf{b}) = \left(g_i W_i g_{i-1}^{-1} \ , \ g_i b_i\right)_{i=1}^{L}. \tag{A.8}$$

where $g_0 = \mathrm{id}_{n_0}$ and $g_L = \mathrm{id}_{n_L}$. This action amounts to changing the basis at each hidden feature space. The Lie algebra of $\mathrm{GL}_{\mathbf{n}^{\text{hidden}}}$ is

$$\mathfrak{gl}_{\mathbf{n}^{\text{hidden}}} = \mathfrak{gl}_{n_1} \times \cdots \times \mathfrak{gl}_{n_{L_1}},$$

and the infinitesimal action of the tuple $M \in \mathfrak{gl}_{\mathbf{n}^{\text{hidden}}}$ is given by:

$$\mathfrak{gl}_{\mathbf{n}^{\text{hidden}}} \circlearrowright \mathsf{Param}(\mathbf{n}) \qquad \mathbf{M} \cdot (\mathbf{W}, \mathbf{b}) \mapsto (M_i W_i - W_i M_{i-1} \ , \ M_i b_i)_{i=1}^{L}$$

where we set M_0 and M_L to be the zero matrices.

Example A.2.2. *In the case $L = 2$, with dimension vector $\mathbf{n} = (n, h, m)$ and no biases. The hidden symmetry group is GL_h with Lie algebra $\mathfrak{gl}_h$. The action of the group and the infinitesimal action of the Lie algebra are given by:*

$$\mathrm{GL}_h \circlearrowright \mathsf{Param}(n,h,m) = \mathbb{R}^{h \times n} \times \mathbb{R}^{m \times h} \qquad g \cdot (V, U) = \left(gV, U g^{-1}\right)$$

$$\mathfrak{gl}_h \circlearrowright \mathsf{Param}(n,h,m) = \mathbb{R}^{h \times n} \times \mathbb{R}^{m \times h} \qquad M \cdot (V, U) = (MV, -UM)$$

A.2.5 Linear symmetries

Consider a feedforward fully-connected neural network with widths $\mathbf{n} = (n_0, \ldots, n_L)$, so that the parameters space consists of tuples of weights and biases $\boldsymbol{\theta} = (W_i \in \mathbb{R}^{n_i \times n_{i-1}}, b_i \in \mathbb{R}^{n_i})_{i=1}^{L}$. For each hidden layer $0 < i < L$, let G_i be a subgroup of GL_{n_i}, and let $\pi_i : G_i \to \mathrm{GL}_{n_i}(\mathbb{R})$ be a representation (in many cases, we take $\pi_i(g) = g$). Hence the product $G = G_1 \times G_{L-1}$ is a subgroup of the hidden symmetry group $\mathrm{GL}_{\mathbf{n}^{\text{hidden}}}$. Define an action of G on Param via

$$\forall g = (g_1, \ldots, g_L) \in G, \qquad g \cdot W_i = g_i W_i \pi_{i-1}(g_{i-1}^{-1}) \qquad g \cdot b_i = g_i b_i \tag{A.9}$$

where g_0 and g_L are the identity matrices id_{n_0} and id_{n_L}, respectively. This is a version of the action defined in equation A.8, with the addition of the twists resulting from the representations π_i.

We now consider the resulting infinitesimal action. For each i, the representation π_i induces a Lie algebra representation $d(\pi_i) : \mathfrak{g}_i \to \mathfrak{gl}_{n_i}$. The infinitesimal action of the Lie algebra $\mathfrak{g} = \mathfrak{g}_1 \times \cdots \times \mathfrak{g}_L$ induced by A.9 is given by:

$$\forall M = (M_1, \ldots, M_L) \in \mathfrak{g}, \qquad M \cdot W_i = M_i W_i - W_i d(\pi_{i-1})(M_{i-1}) \qquad M \cdot b_i = M_i b_i \tag{A.10}$$

The proof of the first part of the following Proposition proceeds by induction, where the key computation is that of from equation 2.4. The second part relies on equation 2.6.

Proposition A.2.3. *Suppose that, for each $i = 1, \ldots, L$, the activation σ_i intertwines the two actions of G_i, that is, $\sigma_i(g_i z_i) = \pi_i(g_i)\sigma(z_i)$ for all $g_i \in G_i$, $z_i \in \mathbb{R}^{n_i}$. Then:*

1. *(Combined equivariance of activations) The action of $G = G_1 \times \cdots \times G_L$ defined in A.9 is a symmetry of the parameter space.*

2. *(Infinitesimal equivariant action) The action of $\mathfrak{g} = \mathfrak{g}_1 \times \cdots \times \mathfrak{g}_L$ defined in A.10 satisfies $\langle \nabla_{\boldsymbol{\theta}} L, M \cdot \boldsymbol{\theta} \rangle$ for all $\boldsymbol{\theta} \in \mathsf{Param}$ and all $M \in \mathfrak{g}$.*

Proof. Let $\mathbf{g} = (g_i)_{i=1}^{L-1} \in G$, so that $g_i \in G_i \subseteq \mathrm{GL}_{n_i}$. As usual, set $g_0 = \mathrm{id}_{n_0}$ and $g_L = \mathrm{id}_{n_L}$. Also set π_0 and π_L to be the identity maps on GL_{n_0} and GL_{n_L}, respectively. Fix parameters $\boldsymbol{\theta}$ and an input value $x \in \mathbb{R}^n$. We show by induction that the following relation between the partial feedforward functions holds:

$$F_{\mathbf{g}\cdot\boldsymbol{\theta},i}(x) = \pi_i(g_i)(F_{\boldsymbol{\theta},i}(x))$$

for $i = 0, \ldots, L$. The base step is trivial. For the induction step, we use the recursive definition of the partial feedforward functions:

$$\begin{aligned} F_{\mathbf{g}\cdot\boldsymbol{\theta},i}(x) &= \sigma_i(g_i W_i \pi_{i-1}(g_{i-1}^{-1}) F_{\mathbf{g}\cdot\boldsymbol{\theta},i-1}(x) + g_i b_i) \\ &= \sigma_i(g_i(W_i \pi_{i-1}(g_{i-1}^{-1})\pi_{i-1}(g_{i-1}) F_{\boldsymbol{\theta},i-1}(x) + b_i) \\ &= \pi_i(g_i)\sigma_i(W_i F_{\boldsymbol{\theta},i-1}(x) + b_i) = \pi_i(g_i) F_{\boldsymbol{\theta},i}(x) \end{aligned}$$

Hence $\boldsymbol{\theta}$ and $g \cdot \boldsymbol{\theta}$ define the same feedforward function. Since the loss function depends on the parameters only through the feedforward function, the first claim follows. The second claim is a consequence of Proposition 2.3.9. □

Note that two-layer case of the above result amounts to equation 2.4 (the argument can be simplified in that case). While Proposition A.2.3 is stated for feedforward networks, it can easily be adopted to more general settings, such as networks with skip connections and quiver neural networks. Denoting the Jacobian of $\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ at $z \in \mathbb{R}^{n_i}$ by $d(\sigma_i)_z \in \mathbb{R}^{n_i \times n_i}$, the infinitesimal form of version of $\sigma_i(gz) = \pi_i(g)\sigma_i(z)$ is

$$\langle d(\sigma_i)_z, Mz \rangle = d\pi_i(M)\sigma_i(z) \qquad \forall M \in \mathfrak{gl}_{n_i} \tag{A.11}$$

When the activation is pointwise, we have $Mz \odot \sigma_i'(z) = d(\pi)_i(M)\sigma_i(z)$, where $\odot$ denotes elementwise multiplication. We now illustrate Proposition A.2.3 through examples in the two-layer case with input dimension n, hidden dimension h, hidden activation σ, and output

dimension m.

Example A.2.4 (Linear networks)**.** *For linear networks, we have $\sigma(x) = x$. One can take $\pi(g) = g$ and $G = \mathrm{GL}_h(\mathbb{R})$.*

Example A.2.5 (Homogeneous activations)**.** *Suppose the activation $\sigma : \mathbb{R}^h \to \mathbb{R}^h$ is homogeneous, so that (1) σ is applied pointwise in the standard basis, and (2) t there exists $\alpha > 0$ such that $\sigma(cz) = c^\alpha \sigma(z)$ for all $c \in \mathbb{R}_{>0}$ and $z \in \mathbb{R}^h$. These σ are equivariant under the positive scaling group $G \subset \mathrm{GL}_h$ consisting of diagonal matrices with positive diagonal entries. For $g \in G$, we have $g = \mathrm{diag}(\mathbf{c})$ is a diagonal matrix with $\mathbf{c} = (c_1, \ldots, c_h) \in \mathbb{R}^h_{>0}$. For $z = (z_1, \ldots, z_h) \in \mathbb{R}^h$, we have $\sigma(gz) = \sum_j \sigma(c_j z_j) = \sum_j c_j^\alpha \sigma(z_j) = g^\alpha \sigma(z)$. Hence, the equivariance condition holds with $\pi(g) = g^\alpha$. Since $d\pi(M) = \alpha M$ for any element M of the Lie algebra $\mathfrak{g}$ of G, the infinitesimal version of rescaling invariance of homogeneous σ becomes $Mz \odot \sigma'(z) = \alpha M \sigma(z)$.*

Example A.2.6 (LeakyReLU)**.** *This is a special case of homoegeneous activation, defined as $\sigma(z) = \max(z, 0) - s\min(z, 0)$, with $s \in \mathbb{R}_{>0}$. We have $\alpha = 1$, and $\pi(g) = g$. Since $\sigma(z) = z\sigma'(z)$, infinitesimal equivariance becomes $Mz \odot \sigma'(z) = M\sigma(z)$.*

Example A.2.7 (Radial rescaling activations)**.** *A less trivial example of continuous symmetries is the case of a radial rescaling activation [51] where for $z \in \mathbb{R}^h \setminus \{0\}$, $\sigma(z) = f(\|z\|)z$ for some function $f : \mathbb{R} \to \mathbb{R}$. Radial rescaling activations are equivariant under rotations of the input: for any orthogonal transformation $g \in O(h)$ (that is, $g^T g = I$) we have $\sigma(gz) = g\sigma(z)$ for all $z \in \mathbb{R}^h$. Indeed, $\sigma(gz) = f(\|gz\|)(gz) = g(f(\|z\|)z) = g\sigma(z)$, where we use the fact that $\|gz\| = z^T g^T gz = z^T z = \|z\|$ for $g \in O(h)$. Hence, equation 2.4 is satisfied with $\pi(g) = g$.*

A.2.6 Linear symmetries lead to extended, flat minima

In this section, we show that, in the case of a linear group action, applying the action of any element of the group to a local minimum yields another local minimum. This fact is a corollary of a more general result; in order to describe it and remove ambiguity, we include the following clarifications. Let G be a matrix Lie group acting as a linear symmetry. Fix a basis

$(\theta_1, \ldots, \theta_d)$ of the parameter space. The gradient $\nabla_{\boldsymbol{\theta}} L$ of the loss L at a point $\boldsymbol{\theta} \in \mathsf{Param}$ in the parameter space as another vector in $\mathsf{Param} \simeq \mathbb{R}^d$, whose i-th coordinate is the partial derivative $\frac{\partial L}{\partial \theta_i}\big|_{\boldsymbol{\theta}}$. Hence, it makes sense to apply the group action to the gradient: $g \cdot \nabla_{\boldsymbol{\theta}} L$. We regard vectors in $\mathsf{Param} \simeq \mathbb{R}^d$ as column vectors with d rows. Thus, the transpose of any vector is a row vector with d columns. In the case of the gradient, its transpose at $\boldsymbol{\theta}$ matches the *Jacobian* $dL_{\boldsymbol{\theta}} \in \mathbb{R}^{1\times d}$ of L (see Appendix A.2.1), that is: $dL_{\boldsymbol{\theta}} = (\nabla_{\boldsymbol{\theta}} L)^T$. Alternative notation for the Jacobian is $dL_{\boldsymbol{\theta}_0} = \frac{\partial L}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\theta}_0}$, where we now use $\boldsymbol{\theta}$ as a dummy variable and $\boldsymbol{\theta}_0 \in \mathsf{Param}$ as a specific value. As noted above, we are interested in matrix Lie groups $G \subseteq \mathrm{GL}_d(\mathbb{R}) = \mathrm{GL}(\mathsf{Param})$, and assume that the matrix transpose g^T belongs to G for any $g \in G$. These assumptions hold in all examples of interest. We have the following reformulation of Proposition 2.3.3:

Proposition A.2.8. *Suppose the action of G on the parameter space is linear and leaves the loss invariant. Then the gradients of L at any $\boldsymbol{\theta}_0$ and $g \cdot \boldsymbol{\theta}_0$ are related as follows:*

$$g^T \cdot \nabla_{g\cdot\boldsymbol{\theta}_0} L = \nabla_{\boldsymbol{\theta}_0} L \qquad \forall g \in G,\ \forall \boldsymbol{\theta}_0 \in \mathsf{Param} \tag{A.12}$$

If $\boldsymbol{\theta}^*$ is a critical point (resp. local minimum) of L, then so is $g \cdot \boldsymbol{\theta}^*$.

Sketch of proof. Let $T_g : \mathsf{Param} \to \mathsf{Param}$ be the transformation corresponding to $g \in G$. The the Jacobian $dL_{\boldsymbol{\theta}_0}$ is given by:

$$dL_{\boldsymbol{\theta}_0} = \frac{\partial L}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}_0} = \frac{\partial (L \circ T_g)}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}_0} = \frac{\partial L}{\partial \boldsymbol{\theta}}\bigg|_{g\cdot\boldsymbol{\theta}_0} \frac{\partial T_g}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}_0} = dL_{g\cdot\boldsymbol{\theta}_0} \circ T_g$$

where we use the definition of the Jacobian, the invariance of the loss ($L \circ T_g = L$), the chain rule, and the linearity of the action. The result follows from applying $T_{g^{-1}}$ on the right to both sides, and taking transposes (see Appendix A.2.1). The last statement follows from the invariance of L under the action of G, and the fact that $\nabla_{\boldsymbol{\theta}^*} L = 0$ at a critical point $\boldsymbol{\theta}^*$ of L. □

We conclude that, if $\boldsymbol{\theta}^*$ is a critical point, then the set $\{g \cdot \boldsymbol{\theta} \mid g \in G\}$ belongs to the critical locus. This set is known as the *orbit* of $\boldsymbol{\theta}$ under the action of G, and is isomorphic to the

quotient $G/\mathrm{Stab}_G(\boldsymbol{\theta})$, where $\mathrm{Stab}_G(\boldsymbol{\theta}) = \{g \in G \mid g \cdot \boldsymbol{\theta} = \boldsymbol{\theta}\}$ is the *stabilizer* subgroup of $\boldsymbol{\theta}$ in G. In the case of a linear action, the orbit is a smooth manifold. While the results above imply that the critical locus is a union of G-orbits, they do not imply, in general, that the critical locus is a single G-orbit. They also do not rule out the case that the stabilizer is a somewhat 'large' subgroup of G, in which case the orbit would have low dimension. However, in many cases, there is a topologically dense subset of parameter values $\boldsymbol{\theta} \in \mathsf{Param}$ whose orbits all have the same dimension. We call such an orbit a 'generic' orbit. We now turn our attention to examples of two-layer networks where such a generic orbit exists.

Flat directions in the two-layer case

Recall that the parameter space of a two-layer network is $\mathsf{Param} = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$, where the dimension vector is (m,h,n), and we write elements as (U,V). The action of GL_h is $g \cdot (U,V) = (Ug^{-1}, gV)$. Let $\mathsf{Param}^\circ \subseteq \mathsf{Param}$ be the subset of pairs (U,V) where each of U and V have full rank. This is an open dense subset of Param, and is preserved by the GL_h-action.

Proposition A.2.9. *The* GL_h*-orbit of each element of* Param° *has dimension*

$$\dim(\mathrm{Orbit}) = h^2 - \max(0,h-n)\max(0,h-m).$$

Proof. Fix $(U,V) \in \mathsf{Param}^\circ$. Suppose $h \leq n$, so that $h^2 - \max(0,h-n)\max(0,h-m) = h^2$. Then $V \in \mathbb{R}^{h\times n}$ defines a surjective linear map, so has a right inverse $V^\dagger \in \mathbb{R}^{n\times h}$. If $g \in \mathrm{GL}_h$ belongs to the stabilizer of (U,V), then we have $gV = V$. Applying $V^\dagger$ on the right to both sides, we obtain $g = \mathrm{id}_h$. Thus, the stabilizer of (U,V) is trivial, and the orbit has dimension equal to the dimension of the group, namely h^2. The case $h \leq m$ is similar.

Now suppose $h > \max(n,m)$. In this case, $h^2 - \max(0,h-n)\max(0,h-m) = h(n+m) - nm$. Set $U_0 = \begin{bmatrix} \mathrm{id}_m & 0 \end{bmatrix} \in \mathbb{R}^{m\times h}$ and $V_0 = \begin{bmatrix} \mathrm{id}_n \\ 0 \end{bmatrix} \in \mathbb{R}^{h\times n}$, so that the last $h-m$ rows of U_0 are zero, and the last $h-n$ columns of V_0 are zero. Then, by the rank assumption, there exists

$g_1 \in \mathrm{GL}_h$ such that $Ug_1 = U_0$, and there exists $g_2 \in \mathrm{GL}_h$ such that $g_2^{-1}V = V_0$. Without loss of generality, we can take g_1 and g_2 such that $\det(g_1) > 0$ and $\det(g_2) > 0$. Thus, both g_1 and g_2 belong to the component of the identity in GL_h.

Next, consider the action of $G = \mathrm{GL}_h$ on full rank matrices in $\mathbb{R}^{m\times h}$ and $\mathbb{R}^{h\times n}$ individually. We have that $\mathrm{Stab}_G(U) = g_1\mathrm{Stab}_G(U_0)g_1^{-1}$ and $\mathrm{Stab}_G(V) = g_2\mathrm{Stab}_G(V_0)g_2^{-1}$. The stabilizer in GL_h of the pair $(U,V) \in \mathsf{Param}^\circ$ can be written as:

$$\mathrm{Stab}_G(U,V) = \{g \in G \mid Ug^{-1} = U \text{ and } gV = V\} \tag{A.13}$$

$$= \mathrm{Stab}_G(U) \cap \mathrm{Stab}_G(V) \tag{A.14}$$

$$= (g_1\mathrm{Stab}_G(U_0)g_1^{-1}) \cap (g_2\mathrm{Stab}_G(V_0)g_2^{-1}) \tag{A.15}$$

Since g_1 and g_2 belong to the connected component of the identity, the dimension of the intersection $(g_1\mathrm{Stab}_G(U_0)g_1^{-1}) \cap (g_2\mathrm{Stab}_G(V_0)g_2^{-1})$ is equal to the dimension[3] of $\mathrm{Stab}_G(U_0) \cap \mathrm{Stab}_G(V_0)$. Hence we reduce the problem to computing the dimension of $\mathrm{Stab}_G(U_0) \cap \mathrm{Stab}_G(V_0)$.

To this end, observe that a matrix g belongs to $\mathrm{Stab}_G(U_0)$ (resp. $\mathrm{Stab}_G(V_0)$) if and only if is of the form:

$$g = \begin{bmatrix} \mathrm{id}_m & 0 \\ * & * \end{bmatrix} \qquad (\text{resp.} \quad g = \begin{bmatrix} \mathrm{id}_n & * \\ 0 & * \end{bmatrix})$$

where the lower left $* \in \mathbb{R}^{(h-m)\times m}$ and the lower right $* \in \mathrm{GL}_{h-m}$ (resp. upper right $* \in \mathbb{R}^{n\times(h-n)}$ and lower right $* \in \mathrm{GL}_{h-n}$) are arbitrary. If $m \geq n$, taking the intersection amounts to considering matrices of the form:

$$g = \begin{bmatrix} \mathrm{id}_n & 0 & 0 \\ 0 & \mathrm{id}_{m-n} & 0 \\ 0 & * & * \end{bmatrix}$$

where the rows and columns are divided according to the partition $h = n + (m-n) + (h-m)$. If

[3]Explicitly, fix a continuous path $\gamma_i : [0,1]$ such that $\gamma_i(0)$ is the identity in G and $\gamma_i(1) = g_i$, for $i = 1,2$. The dimension of $(\gamma_1(t)\mathrm{Stab}_G(U_0)\gamma_1(t)^{-1}) \cap (\gamma_2(t)\mathrm{Stab}_G(V_0)\gamma_2(t)^{-1})$ is constant along this path.

$m \geq n$, taking the intersection amounts to considering matrices of the form:

$$g = \begin{bmatrix} \mathrm{id}_m & 0 & 0 \\ 0 & \mathrm{id}_{n-m} & * \\ 0 & 0 & * \end{bmatrix}$$

where the rows and columns are divided according to $h = m + (n-m) + (h-n)$. In both cases, the dimension of the intersection is $(h-n)(h-m) = h^2 - hn - hm + nm$. We obtain the dimension of the orbit as: $h^2 - (h-n)(h-m) = h(n+m) - nm$. □

Recall that the symmetry group for homogenous activations is the coordinate-wise positive rescaling subgroup of GL_h, consisting of diagonal matrices with positive entries along the diagonal. We denote this subgroup as $T_+(h)$. Similarly, the symmetry group for radial rescaling activation is the orthogonal group $O(h)$. For linear networks, the activation is the identity function, so the symmetry group is all of GL_h.

Corollary A.2.10. *The orbit of a point in* Param° *under the appropriate symmetry group is given by:*

Type of activation	Symmetry group	Dimension of generic orbit
Linear	GL_h	$h^2 - \max(0, h-n)\max(0, h-m)$
Homogeneous	$T_+(h)$	$\min(h, \max(n,m))$
Radial rescaling	$O(h)$	$\begin{cases} \binom{h}{2} & \text{if } h \leq \max(n,m) \\ \binom{h}{2} - \binom{h-\max(m,n)}{2} & \text{otherwise} \end{cases}$

Proof. Adopt the notation of the proof of the above Proposition. The stabilizer in $T_+(h)$ of (U_0, V_0) is the intersection of the stabilizer in GL_h of (U_0, V_0) and $T_+(h)$. This intersection is easily seen to have dimension $\max(0, h - \max(n,m))$. Subtracting this from $\dim(T_+(h)) = h$,

we obtain the result for the homogeneous case. For the orthogonal case, the stabilizer in $O(h)$ of (U_0, V_0) is the intersection of the stabilizer in GL_h of (U_0, V_0) and $O(h)$. This intersection has dimension 0 if $h \leq \max(n,m)$ and $\binom{h-\max n,m}{2}$ otherwise. Subtracting from $\dim(O(h)) = \binom{h}{2}$, we obtain the result for the radial rescaling case. □

A.2.7 Conserved quantities

We now turn our attention to gradient flow and conserved quantities. In this section, we give a formal definition of a conserved quantity. Let $\mathscr{V} = \mathbb{R}^d$ be the standard vector space of dimension d. Suppose $L : \mathscr{V} \to \mathbb{R}$ is a differentiable function. Let $\mathrm{Flow}_t : \mathscr{V} \to \mathscr{V}$ be the flow for time t along the reverse gradient vector field, so that:

$$\left.\frac{d}{dt}\right|_0 [t \mapsto \mathrm{Flow}_t(v)] = -\nabla_v L$$

Note that Flow_0 is the identity on $\mathscr{V}$, and, for any s,t, the composition of Flow_s and Flow_t is Flow_{s+t}. We will write $v(t)$ for $\mathrm{Flow}_t(v)$, so that $\dot{v}(s) = -\nabla_{v(s)} L$. A *conserved quantity* is a function $Q : \mathscr{V} \to \mathbb{R}$ that satisfies either of the following equivalent conditions:

1. For any t, we have $Q(v(t)) = Q(v)$.
2. Let $\dot{Q}(v) = \frac{d}{dt}\big|_0 [t \mapsto Q(v(t))]$ be the derivative of Q along the flow. Then $\dot{Q} \equiv 0$.
3. The gradients of Q and L are point-wise orthogonal, that is, $\langle \nabla_v Q, \nabla_v L \rangle = 0$ for all $v \in \mathscr{V}$.

The equivalence of (1) and (2) are immediate. To show the equivalence of the third and second statements, let $v \in \mathscr{V}$ and compute:

$$\langle \nabla_v Q, \nabla_v L \rangle = dQ_{v(0)} \circ \nabla_v L = dQ_{v(0)} \circ \left.\frac{d}{dt}\right|_0 v(t) = \frac{d}{dt}\left(Q \circ v(t)\right),$$

where we use the definition of the flow in the second equality, and the chain rule in the third.

We note that, if $f : \mathbb{R} \to \mathbb{R}$ is any function, and Q is a conserved quantity, the $f \circ Q$ is also a conserved quantity. Additionally, any linear combination of conserved quantities is again a conserved quantity. Let $\text{Conserv}(\mathscr{V}, L)$ denote the vector space of conserved quantities for the gradient flow of $L : \mathscr{V} \to \mathbb{R}$. For any $v \in \mathscr{V}$, there a map:

$$\nabla_v : \text{Conserv}(\mathscr{V}, L) \to T_v \mathscr{V} = \mathbb{R}^d, \qquad Q \mapsto \nabla_v Q$$

taking a conserved quantity to the value of its gradient at v. By the above discussion, the map ∇_v is valued in the kernel of the differential dL_v.

A.2.8 Conserved quantities from a group action

Let G be a subgroup of the general linear group $\text{GL}_d(\mathbb{R})$. Thus, there is a linear action of G on $\mathscr{V} = \mathbb{R}^d$. Suppose the function L is invariant for the action of G, that is,

$$L(g \cdot v) = L(v) \qquad \forall v \in \mathscr{V} \quad \forall g \in G.$$

Let $\mathfrak{g} = \text{Lie}(G)$ be the Lie algebra of G, which is a Lie subalgebra of $\mathfrak{gl}_d = \mathbb{R}^{d \times d}$. The *infinitesimal action* of $\mathfrak{g}$ on $\mathscr{V}$ is given by $\mathfrak{g} \times V \to V$, taking (M, v) to Mv.

Proposition A.2.11. *Let $L : \mathscr{V} \to \mathbb{R}$ be a G-invariant function, and let $M \in \mathfrak{g}$.*

1. *For any $v \in \mathscr{V}$, the gradient of L and the infinitesimal action of M are orthogonal:*

$$\langle \nabla_v L, Mv \rangle = 0.$$

2. *Suppose $\gamma : (a, b) \to \mathscr{V}$ is a gradient flow curve for L. Then:*

$$(\dot{\gamma}(t))^T M \gamma(t) = 0 \qquad \forall t \in (a, b)$$

3. *Suppose M^T belongs to $\mathfrak{g}$. Then the function*

$$Q_M : \mathscr{V} \to \mathbb{R}, \qquad v \mapsto v^T M v$$

is a conserved quantity for the gradient flow of L.

Proof. For the first claim, observe that the invariance of L implies that the left diagram commutes:

$$\begin{array}{ccccc} G & \xrightarrow{\text{inc}} & \mathbb{R}^{d\times d} & \xrightarrow{\text{ev}_v} & \mathbb{R}^d \\ \downarrow & & & & \downarrow L \\ \{v\} & & \xrightarrow[L]{} & & \mathbb{R} \end{array} \qquad \begin{array}{ccccc} \mathfrak{g} & \xrightarrow{d\text{inc}_1} & \mathbb{R}^{d\times d} & \xrightarrow{d(\text{ev}_x)_{\text{id}_d}} & \mathbb{R}^d \\ \downarrow & & & & \downarrow dL_v \\ 0 & & \longrightarrow & & \mathbb{R} \end{array}$$

where $\text{inc} : G \to \mathbb{R}^{d\times d}$ is the inclusion (which passes through the inclusion of G in to $\text{GL}_d(\mathbb{R})$), $\text{ev}_v : \mathbb{R}^{d\times d} \to \mathbb{R}^d$ be the evaluation map at v, and the left vertical map is the constant map at $\{v\}$. Indeed, the clockwise composition is $g \mapsto L(gv)$, which is equal to the constant map at $g \mapsto L(v)$. The chain rule implies that taking Jacobians at the identity 1 of G results in the commutative diagram on the right. where $\mathfrak{g}$ is the Lie algebra of G, identified with the tangent space of G at the identity; the tangent space of the vector space $\mathbb{R}^d$ at v is canonically identified with $\mathbb{R}^d$; and the the zero appears because the tangent space of a single point is zero. The derivative of the inclusion map is the inclusion $\mathfrak{g} \hookrightarrow \mathfrak{gl}_d = \mathbb{R}^{d\times d}$, while the derivative the evaluation map is itself as it is a linear map. Hence, for $M \in \mathfrak{g}$, we have:

$$0 = dL_v \circ d(\text{ev}_v)_{\text{id}_d} \circ d\text{inc}_1(M) = dL_v \circ \text{ev}_v(M) = dL_v(Mv) = \langle \nabla_v L, Mv \rangle.$$

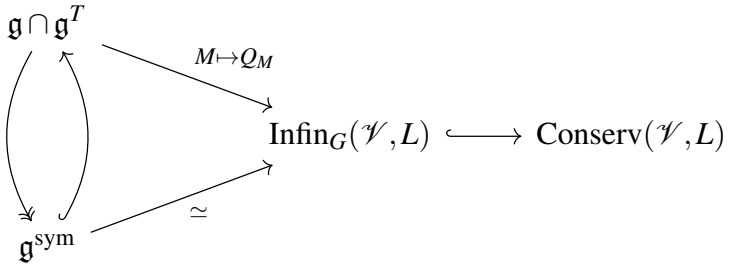
The first claim follows. The second claim is consequence of the first claim, together with the definition of a gradient flow curve. For the third claim, we take the derivative of the composition of Q_M with a gradient flow curve γ:

$$\frac{d}{dt}(Q_M \circ \gamma) = (\dot{\gamma}(t))^T (M + M^T)\gamma(t)$$

$$
\begin{aligned}
&= \langle \dot{\gamma}(t), (M + M^T)\gamma(t) \rangle \\
&= -\langle \nabla_{\gamma(t)} L, (M + M^T)\gamma(t) \rangle \\
&= -\langle \nabla_{\gamma(t)} L, M\gamma(t) \rangle - \langle \nabla_{\gamma(t)} L, M^T\gamma(t) \rangle
\end{aligned}
$$

Both terms in the last expression are constantly zero by the second claim. Hence Q_M is constant on any gradient flow curve, and so it is a conserved quantity. □

We summarize some of the results and constructions of this section diagrammatically. Let $\mathfrak{g}^{\text{sym}}$ denote the vector space of symmetric matrices in $\mathfrak{g}$ (this is not a Lie subalgebra in general). Observe that $\mathfrak{g} \cap \mathfrak{g}^T$ is the set of all $M \in \mathfrak{g}$ such that $M^T \in \mathfrak{g}$. Let $\text{Infin}_G(\mathscr{V}, L)$ denote the vector space of infinitesimal-action conserved quantities for the gradient flow of the G-invariant function $L : \mathscr{V} \to \mathbb{R}$. We have:



where the map $\mathfrak{g} \cap \mathfrak{g}^T \to \mathfrak{g}^{\text{sym}}$ takes M to its symmetric part $\frac{1}{2}\left(M + M^T\right)$, while the map $\mathfrak{g}^{\text{sym}} \hookrightarrow \mathfrak{g} \cap \mathfrak{g}^T$ is the natural inclusion. We note that $\mathfrak{g} \cap \mathfrak{g}^T$ is the Lie algebra of the group $G \cap G^T$, while $\mathfrak{g}^{\text{sym}}$ is in general not a Lie algebra. By definition, the vector space $\text{Infin}_G(\mathscr{V}, L)$ is the image of the map $M \to Q_M$ defined on $\mathfrak{g} \cap \mathfrak{g}^T$. It is straightforward to verify the following result:

Corollary A.2.12. *The map $M \to Q_M$ establishes an isomorphism of vector spaces: $\mathfrak{g}^{\text{sym}} \xrightarrow{\sim} \text{Infin}_G(\mathscr{V}, L)$.*

As discussed in Section 2.3.2, applying a symmetry g to a minimum $\boldsymbol{\theta}^*$ of L yields another minimum $g \cdot \boldsymbol{\theta}^*$. Using flattened $\boldsymbol{\theta} \in \mathbb{R}^d$ it is easy to show that acting with g changes some $Q_M(\boldsymbol{\theta}) = \boldsymbol{\theta}^T M \boldsymbol{\theta}$. Let $g = \exp_{M'}(t) \approx I + tM'$, with $M' \in \mathfrak{g}$ and $0 < \eta \ll 1$ be a $g \in G$ close to

identity. We have $Q_M(g \cdot \boldsymbol{\theta}) = Q_M + t\boldsymbol{\theta}^T \left(M'^T M + MM' \right) \boldsymbol{\theta} + O(\eta^2)$. Thus, whenever $M'^T M + MM' \neq 0$, applying g changes the value of Q_M. Therefore, Q_M can be used to parameterize the flat minima. However, for anti-symmetric M, we could not find nonzero Q explicitly.

Anti-symmetric case

Suppose $M \in \mathfrak{g}$ is anti-symmetric, so $M = -M^T$. Let $\gamma : (a,b) \to \mathbb{R}^d$ be a gradient flow curve. Write γ in coordinates as $\gamma = (\gamma_1, \ldots, \gamma_d)$. Proposition A.2.11 implies that $(\dot{\gamma}(t))^T M\gamma = 0$. Hence we have:

$$0 = \sum_{i<j} m_{ij} \left(\dot{\gamma}_i \gamma_j - \gamma_i \dot{\gamma}_j \right) = \sum_{i<j} m_{ij} \gamma_i^2 \left(\frac{\gamma_j}{\gamma_i} \right)' = \sum_{i<j} m_{ij} r_{ij}^2 \dot{\theta}_{ij}$$

where $r_{ij} = r_{ij}(t)$ is equal to $\sqrt{\gamma_i(t)^2 - \gamma_j(t)^2}$ and $\theta_{ij} = \theta_{ij}(t)$ is the angle between the i-th coordinate axis and the ray from the origin to the projection of $\gamma(t)$ to the (i,j)-plane. One verifies the last equality using the definition of θ_{ij} in terms of the arctangent of the quotient γ_j/γ_i. We see that $(r_{ij}(t), \theta_{ij}(t))$ are the polar coordinates for the point $(\gamma_i(t), \gamma_j(t)) \in \mathbb{R}^2$.

Case $d = 2$.

Then $M = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$ for some nonzero $a \in \mathbb{R}$, and so:

$$0 = \dot{\gamma}^T M\gamma = a\dot{\gamma}_1(t)\gamma_2(t) - a\gamma_1(t)\dot{\gamma}_2(t) = ar(t)^2\dot{\theta}(t)$$

where $r(t)$ and $\theta(t)$ are polar coordinates. Setting the final expression equal to zero, we obtain that $\theta(t)$ is constant along any flow line $\gamma(t)$ that begins away from the origin.

Case $d = 3$.

Then $M = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix}$ for some $a, b, c \in \mathbb{R}$, and so:

$$0 = \dot{\gamma}^T M \gamma = a r_{12}^2 \dot{\theta}_{12} + b r_{13}^2 \dot{\theta}_{13} + c r_{23}^2 \dot{\theta}_{23}$$

A.2.9 Examples of conserved quantities for neural networks

We now compute conserved quantities for gradient flow on neural network parameter spaces in the case of linear, homogeneous, and radial networks. In each case, we state results first in the for a general multi-layer network, and then for the running example of a two-layer network. Throughout, $\odot$ denotes the Hadamard product of matrices, defined by entrywise multiplication. We also set $\tau(M)$ to be the sum of all entries in a matrix M. We note that, for square matrices M and N of the same size, $\tau(M \odot N) = \mathrm{Tr}(M^T N)$, which is the same as the inner product of the flattened versions of M and N.

The notation for the running example of a two-layer network is as follows. We set the input and output dimensions both equal to one, hidden dimension equal to two, and no bias vectors. The hidden layer activation is $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$. The parameter space is $\mathsf{Param} = \mathbb{R}^{2\times 1} \times \mathbb{R}^{1\times 2}$, with elements written as a pair of matrices: $(U, V) = \left(\begin{bmatrix} u_1 & u_2 \end{bmatrix}, \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \right)$. The hidden symmetry group is GL_2, with action given by:

$$\mathrm{GL}_2(\mathbb{R}) \times \mathsf{Param} \to \mathsf{Param}, \qquad (g, U, V) \mapsto g \cdot (U, V) = \left(U g^{-1}, g V \right).$$

The Lie algebra $\mathfrak{gl}_2$ of $\mathrm{GL}_2(\mathbb{R})$ consists of all two-by-two matrices.

Conserved quantities for linear networks

Suppose a neural network with L layers has the identity activation $\sigma_i = \mathrm{id}_{n_i}$ in each layer, so that the resulting network is linear. Then it is straightforward to verify that the networks with parameters $\mathbf{g} \cdot (\mathbf{W}, \mathbf{b})$ and $(\mathbf{W}, \mathbf{b})$ have the same feedforward function. Consequently, the loss is invariant for the group action: its value the original and transformed parameters is the same for any choice of training data. (As we will see below, for more sophisticated activations, one needs to restrict to a subgroup of the hidden symmetry group to achieve such invariance.)

Suppose $\mathbf{M} \in \mathfrak{gl}_{\mathbf{n}^{\text{hidden}}}$ is such that $M_i \in \mathfrak{gl}_{n_i}$ is symmetric for each i. The conserved quantity implied by Proposition A.2.11 is:

$$\begin{aligned} Q_{\mathbf{M}}(\mathbf{W}, \mathbf{b}) &= \sum_{i=1}^{L-1} \left(\tau(W_i \odot M_i W_i) + \tau(b_i \odot M_i b_i) - \tau(W_{i+1} \odot W_{i+1} M_i) \right) \\ &= \sum_{i=1}^{L-1} \mathrm{Tr}\left(\left(W_i W_i^T + b_i b_i^T - W_{i+1}^T W_{i+1} \right) M_i \right) \end{aligned}$$

We examine these conserved quantities in the following convenient basis for the space of symmetric matrices in $\mathfrak{gl}_{\mathbf{n}^{\text{hidden}}}$. For $j = 1, \ldots, L$ and $\{k, \ell\} \subseteq \{1, \ldots, n_j\}$, set:

$$E^{(j)}_{\{k,\ell\}} := \begin{cases} E^{(n_j)}_{kk} & \text{if } k = \ell \\ \frac{1}{2}\left(E^{(n_j)}_{k\ell} + E^{(n_j)}_{\ell k} \right) & \text{if } k \neq \ell \end{cases}$$

where $E^{(n_j)}_{k\ell}$ is the elementary $n_j \times n_j$ matrix with the entry in the k-th row and ℓ-th column equal to one, and all other entries equal to zero. Then one computes:

$$Q_{E^{(j)}_{\{k,\ell\}}}(\mathbf{W}, \mathbf{b}) = b^{(j)}_k b^{(j)}_\ell + \sum_{t=1}^{n_{j-1}} w^{(j)}_{kt} w^{(j)}_{\ell t} - \sum_{r=1}^{n_{j+1}} w^{(j+1)}_{rk} w^{(j+1)}_{r\ell}$$

In other words, we take the sum of the following three terms: the product of the k-th and ℓ-th entries of the bias vector b_j, the dot product of the k-th and ℓ-th rows of W_j, and the dot product

of the k-th and ℓ-th columns of W_{j+1}. In particular, we see that every entry of the matrix

$$\mu_i(\mathbf{W},\mathbf{b}) := W_i W_i^T + b_i b_i^T - W_{i+1}^T W_{i+1} \in \mathfrak{gl}_{n_i}$$

is a conserved quantity valued in $\mathfrak{gl}_{n_i}$ rather than in $\mathbb{R}$. Additionally, we have a *moment map*:

$$\mathbf{Q} : \mathsf{Param} \to \mathfrak{gl}^*_{\mathbf{n}^{\text{hidden}}}, \qquad (\mathbf{W},\mathbf{b}) \mapsto \left[\mathbf{M} \mapsto \sum_{i=1}^{L-1} \langle \mu_i(\mathbf{W},\mathbf{b}), M_i\rangle\right]$$

Conserved quantities for linear networks: two-layer case

In the two layer case of a linear network, we have that that the single hidden activation is the identity: $\sigma = \mathrm{id}_2 : \mathbb{R}^2 \to \mathbb{R}^2$. The hidden symmetry group is GL_2 with Lie algebra all of $\mathfrak{gl}_2$. The space of symmetric matrices in $\mathfrak{gl}_2$ is spanned by the matrices:

$$E_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad E_{22} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \text{and} \qquad E_{(1,2)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The corresponding conserved quantities are:

$$Q_{E_{11}}(U,V) = v_1^2 - u_1^2 \qquad Q_{E_{22}}(U,V) = v_2^2 - u_2^2 \qquad Q_{E_{(1,2)}}(U,V) = v_1 v_2 - u_1 u_2$$

Thus, we obtain a three-dimensional space of conserved quantities. (Since GL_2 also contains the orthogonal group $O(2)$, Equation A.16 below holds along any gradient flow curve.)

Conserved quantities for ReLU networks

The pointwise ReLU activation commutes with positive rescaling, so we consider the subgroup of the hidden symmetry group consisting of tuples of diagonal matrices with positive

diagonal entries, that is:

$$G = \{\mathbf{g} \in \mathrm{GL}_{\mathbf{n}^{\text{hidden}}} \mid g_i = \mathrm{Diag}(s_1, \ldots, s_{n_i}) \ , \ s_j > 0\}$$

This subgroup, also known as the positive coordinate-wise rescaling subgroup, is isomorphic to the product $(\mathbb{R}_{>0})^{\sum_{i=1}^{L-1} n_i}$. Its Lie algebra is spanned by the elements $E_{kk}^{(j)}$ defined above, for $j = 1, \ldots, L-1$ and $k = 1, \ldots, n_j$. The conserved quantity implied by Proposition A.2.11 is:

$$Q_{E_{kk}^{(j)}}(\mathbf{W}, \mathbf{b}) = \left(b_k^{(j)}\right)^2 + \sum_{t=1}^{n_{j-1}} \left(w_{kt}^{(j)}\right)^2 - \sum_{r=1}^{n_{j+1}} \left(w_{rk}^{(j+1)}\right)^2$$

In other words, we take the sum of the following three terms: the square of the k-th entry of the bias vector b_j, the norm of the k-th row of W_j, and the norm of the k-th column of W_{j+1}.

Conserved quantities for ReLU networks: two-layer case

In the two-layer case, the positive rescaling group is:

$$G = \left\{ \begin{bmatrix} g_1 & 0 \\ 0 & g_2 \end{bmatrix} \in \mathrm{GL}_2(\mathbb{R}) \mid g_1 \text{ and } g_2 \text{ are positive.} \right\}$$

The Lie algebra of G is the two-dimensional space of diagonal matrices in $\mathfrak{gl}_2$ (with not necessarily positive diagonal entries). In other words, $\mathfrak{g}$ is spanned by the matrices $E_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $E_{22} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. One computes the conserved quantities corresponding to these elements as:

$$Q_{E_{11}}(U, V) = v_1^2 - u_1^2 \qquad Q_{E_{22}}(U, V) = v_2^2 - u_2^2$$

Hence there is a two-dimensional space of conserved quantities coming from the infinitesimal action.

Conserved angular momentum for radial rescaling networks

Suppose each σ_i is a radial rescaling activation $\sigma_i(z) = \lambda_i(|z|)z$, where $\lambda_i : \mathbb{R} \to \mathbb{R}$ is the rescaling factor. Each such activation commutes with orthogonal transformations, so we consider the subgroup of the hidden symmetry group consisting of tuples of orthogonal matrices:

$$G = \{\mathbf{g} \in \mathrm{GL}_{\mathbf{n}^{\text{hidden}}} \mid g_i g_i^T = \mathrm{id}_{n_i} \text{ for all } i\}$$

The Lie algebra of this subgroup consists only of anti-symmetric matrices, and so there are no infinitesimal-action conserved quantities. However, given an anti-symmetric matrix $M_i \in \mathfrak{gl}_{n_i}$ for each i, any gradient flow curve satisfies the following differential equation (encoding conservation of angular momentum):

$$\sum_{i=1}^{L-1} \left(\tau(\dot{W}_i \odot M_i W_i) + \tau(\dot{b}_i \odot M_i b_i) - \tau(\dot{W}_{i+1} \odot W_{i+1} M_i)\right) = 0$$

(cf. Section A.2.8). An equivalent way to write this equation is:

$$\sum_{i=1}^{L-1} \mathrm{Tr}\left(\left(W_i \dot{W}_i^T + b_i \dot{b}_i^T + W_{i+1}^T \dot{W}_{i+1}\right) M_i\right) = 0$$

Indeed, one uses the facts that $\tau(A \odot B) = \mathrm{Tr}(A^T B)$, $\mathrm{Tr}(A^T) = \mathrm{Tr}(A)$, and $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$, for any two matrices A, B of the appropriate size in each case. Using a basis of anti-symmetric matrices, one can show that the matrix

$$\nu_i(\mathbf{W}, \mathbf{b}) := W_i \dot{W}_i^T - \dot{W}_i W_i^T + b_i \dot{b}_i^T - \dot{b}_i b_i^T + W_{i+1}^T \dot{W}_{i+1} - \dot{W}_{i+1}^T W_{i+1} \in \mathfrak{gl}_{n_i}$$

is equal to zero: $\nu_i(\mathbf{W}, \mathbf{b}) = 0$. Note that ν_i depends on taking derivatives with respect to the flow. In fact, ν_i is more properly formulated as a function on the tangent bundle $T(\mathsf{Param})$ of Param, which is then evaluated on the gradient flow vector field. Similarly, we have a moment map $T(\mathsf{Param}) \to \mathfrak{gl}^*_{\mathbf{n}^{\text{hidden}}}$, and the gradient flow vector field is contained in the preimage of zero.

We omit the details.

A basis for the space of anti-symmetric matrices in $\mathfrak{gl}_{\mathbf{n}^{\text{hidden}}}$ is given by:

$$E^{(j)}_{k<\ell} := E^{(n_j)}_{k\ell} - E^{(n_j)}_{\ell k}$$

where $j = 1, \ldots, L-1$, and $k, \ell \in \{1, \ldots, n_j\}$ satisfy $k < \ell$. The differential equation corresponding to $E^{(j)}_{k\leq\ell}$ is given by:

$$r^2_{b_j;k,\ell}\dot{\theta}_{b_j;k,\ell} + \sum_{t=1}^{n_{j-1}} r^2_{W_j;ks,\ell s}\dot{\theta}_{W_j;ks,\ell s} + \sum_{r=1}^{n_{j+1}} r^2_{W_{j+1};rk,r\ell}\dot{\theta}_{W_{j+1};rk,r\ell} = 0$$

where $(r_{b_j;k\ell}, \theta_{b_j;k,\ell})$ are the polar coordinates of the image of b_j under the projection $\mathbb{R}^{n_j} \to \mathbb{R}^2$ which selects only the k-th and ℓ-th coordinates. Similarly, for any pair matrix entries we have a projection $\mathbb{R}^{n_j \times n_{j-1}} \to \mathbb{R}^2$ and can take the polar coordinates of the image of W_j under this projection.

Conserved angular momentum for radial rescaling networks: two-layer case

In the two-layer radial rescaling case, suppose the dimension vector is (n, h, m), and that there are no bias vectors. For $U \in \mathbb{R}^{m\times h}$, $V \in \mathbb{R}^{h\times n}$ and $M \in \mathfrak{so}(\mathfrak{h})$, we have:

$$\begin{aligned}\left\langle \dot{\boldsymbol{\theta}}, M \cdot \boldsymbol{\theta} \right\rangle &= \left\langle (\dot{U}, \dot{V}), (-UM, MV) \right\rangle = -\operatorname{Tr}(\dot{U}^T U M) + \operatorname{Tr}(\dot{V}^T M V) \\ &= \operatorname{Tr}(V\dot{V}^T M) - \operatorname{Tr}(M^T U^T \dot{U}) = \operatorname{Tr}(V\dot{V}^T M) + \operatorname{Tr}(M U^T \dot{U}) \\ &= \operatorname{Tr}(V\dot{V}^T M) + \operatorname{Tr}(U^T \dot{U} M) = \operatorname{Tr}\left[\left(V\dot{V}^T + U^T\dot{U}\right) M\right]\end{aligned}$$

Hence we obtain the differential equation:

$$\operatorname{Tr}\left[\left(V\dot{V}^T + U^T\dot{U}\right) M\right] = 0$$

In the case where $(n,h,m) = (1,2,1)$, we have the two by two orthogonal group:

$$G = O(2) = \left\{ g \in \mathrm{GL}_2(\mathbb{R}) \mid g^T g = \mathrm{id} \right\}$$

The Lie algebra of G consists of anti-symmetric matrices in $\mathfrak{gl}_2$, and contains no non-zero symmetric matrices. Hence, we do not obtain any conserved quantities from the infinitesimal action in this case. However, using the element $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \in \mathfrak{g}$, we obtain that the following differential equation holds along any gradient flow curve:

$$r_U^2 \dot{\theta}_U + r_V^2 \dot{\theta}_V = 0 \tag{A.16}$$

where (r_U, θ_U) are the polar coordinates of $(u_1, u_2) \in \mathbb{R}^2$, and similarly for (r_V, θ_V). Note that the left-hand side of Equation A.16 is a function of t; so if $\gamma : (a,b) \to \mathscr{V}$ is a gradient flow curve, then a more precise version of the equation is $\left(r_U^2 \circ \gamma\right)(t) \cdot (\theta_U \circ \gamma)'(t) + \left(r_V^2 \circ \gamma\right)(t) \cdot (\theta_V \circ \gamma)'(t) = 0$ for all t.

A.2.10 Jacobians: special cases

We conclude this appendix with a side remark on special cases of the Jacobian formalism.

Manifolds.

Suppose M and N are smooth manifolds, and suppose $F : M \to N$ is a smooth map. The differential of F at $m \in M$ is a linear map between the tangent spaces:

$$dF_m : T_m M \to T_{F(m)} N$$

The map dF_m is computed in local coordinate charts as the Jacobian of partial derivatives. If $G : N \to L$ is another smooth map, then the chain rule becomes $d(G \circ F)_m = dG_{F(m)} \circ dF_m$, for any $m \in M$.

Matrix case.

Suppose $L : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a differentiable function. In this case, we regard the Jacobian at $W \in \mathbb{R}^{m\times n}$ as an $n \times m$ matrix:

$$dL_W = \begin{bmatrix} \left.\frac{\partial L}{\partial w_{11}}\right|_W & \left.\frac{\partial L}{\partial w_{21}}\right|_W & \cdots & \left.\frac{\partial L}{\partial w_{m1}}\right|_W \\ \left.\frac{\partial L}{\partial w_{12}}\right|_W & \left.\frac{\partial L}{\partial w_{22}}\right|_W & \cdots & \left.\frac{\partial L}{\partial w_{m2}}\right|_W \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial L}{\partial w_{1n}}\right|_W & \left.\frac{\partial L}{\partial w_{2n}}\right|_W & \cdots & \left.\frac{\partial L}{\partial w_{mn}}\right|_W \end{bmatrix} \in \mathbb{R}^{m\times n}$$

where w_{ij} are the matrix coordinates. If $F : \mathbb{R} \to \mathbb{R}^{m\times n}$ is a differentiable function, we regard its Jacobian at $s \in \mathbb{R}$ as a $m \times n$ matrix:

$$dF_t = \begin{bmatrix} \left.\frac{dF_{11}}{dt}\right|_s & \left.\frac{dF_{12}}{dt}\right|_s & \cdots & \left.\frac{dF_{1n}}{dt}\right|_s \\ \left.\frac{dF_{21}}{dt}\right|_s & \left.\frac{dF_{22}}{dt}\right|_s & \cdots & \left.\frac{dF_{2n}}{dt}\right|_s \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{dF_{m1}}{dt}\right|_s & \left.\frac{dF_{m2}}{dt}\right|_s & \cdots & \left.\frac{dF_{mn}}{dt}\right|_s \end{bmatrix} \in \mathbb{R}^{n\times m}$$

where $F_{ij} : \mathbb{R} \to \mathbb{R}$ are the coordinates of F. Then the chain rule becomes:

$$\left.\frac{d}{dt}\right|_s (L \circ F) = \sum_{i=1}^{m}\sum_{j=1}^{n} \left(\left.\frac{\partial L}{\partial w_{ij}}\right|_{F(s)} \left.\frac{dF_{ij}}{dt}\right|_s \right) = \mathrm{Tr}(dL_{F(s)} \cdot dF_s).$$

In other words, the derivative of the composition $L \circ F$ at $s \in \mathbb{R}$ is the trace of the product of the matrices $dL_{F(s)} \in \mathbb{R}^{n\times m}$ and $dF_s \in \mathbb{R}^{m\times n}$.

A.3 Neural networks: non-linear actions group actions

In this section, we consider a non-linear action of the hidden symmetry group on the parameter space. This action has the advantage that exists for a wider variety of activation

functions (such as the usual sigmoid, which has no linear equivariance properties), and that it is defined for the full general linear group. However, in constrast to the linear action, the non-linear action is data-dependent: the transformation of the weights and biases depends on the input data.

A.3.1 Rotations

We first define certain orthogonal matrices.

Definition A.3.1. *For any tuple of real numbers $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$, define an $(n+1) \times (n+1)$ matrix $R(\boldsymbol{\beta})$ as follows:*

$$(R(\boldsymbol{\beta}))_{ij} = \begin{cases} \cos(\beta_{j-1}) \left(\prod_{k=j}^{i-1} \sin(\beta_k) \right) \cos(\beta_i) & \text{if } j \leq i \\ -\sin(\beta_i) & \text{if } j = i+1 \\ 0 & \text{if } j > i+1 \end{cases}$$

where, by convention, we set $\beta_0 = \beta_{n+1} = 0$.

For example, when $n = 1, 2$, we have:

$$R(\beta) = \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix} \qquad R(\beta_1, \beta_2) = \begin{bmatrix} \cos(\beta_1) & -\sin(\beta_1) & 0 \\ \sin(\beta_1)\cos(\beta_2) & \cos(\beta_1)\cos(\beta_2) & -\sin(\beta_2) \\ \sin(\beta_1)\sin(\beta_2) & \cos(\beta_1)\sin(\beta_2) & \cos(\beta_2) \end{bmatrix}$$

Lemma A.3.2. *For any tuple of real numbers $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$, we have:*

1. $\sum_{i=1}^{n} \cos^2(\beta_i) \prod_{k=1}^{i-1} \sin^2(\beta_k) + \prod_{k=1}^{n} \sin^2(\beta_k) = 1$
2. *The matrix $R(\boldsymbol{\beta})$ is orthogonal.*

Sketch of proof. The first identity follows from a straightforward induction argument, while the proof of the second claim amounts to a computation that invokes the identity of the first claim. □

Proposition A.3.3. *There is a continuous map $R : \mathbb{R}^h \setminus \{0\} \to \mathrm{GL}_h$, written $z \mapsto R_z$, such that:*

1. *For any $z \in \mathbb{R}^h \setminus \{0\}$, the first column of R_z is z. Hence $R_z e_1 = z$, where $e_1 = (1, 0, \ldots, 0)$ is the first basis vector.*
2. *The operator norm of R_z is $\|R_z\| = |z|$.*
3. *If $|z| = 1$, then R_z is an orthogonal matrix.*

Proof. Let $z \in \mathbb{R}^h \setminus \{0\}$, and let $(r, \alpha_1, \ldots, \alpha_{n-1})$ be the (reverse) h-spherical coordinates of z. Hence, $r = |z|$ is the norm of z and the i-th coordinate of z is $z_i = r\left(\prod_{k=1}^{i-1} \sin(\alpha_k)\right)\cos(\alpha_i)$, where $\alpha_h = 0$ by convention. Now set $R_z = |z|R(\alpha_1, \ldots, \alpha_{h-1})$. Using Lemma A.3.2, one concludes that R_z is invertible with inverse $\frac{1}{|z|^2}R_z^T$, so that R_z has operator norm is $|z|$ and R_z is orthogonal if $|z| = 1$. It is also clear that the first column of R_z is equal to z. □

We note the the matrix in A.3.1 has a form similar, but not identical, to the Jacobian matrix for the transformation to n-spherical coordinates. Euler angles provide another way to construct a map $\mathbb{R}^h \setminus \{0\} \to \mathrm{GL}_h$ the same properties as in Proposition A.3.3.

A.3.2 Non-linear action: two-layer case

Consider a two-layer network with dimension vector (m, h, n), no bias vectors, and no output activation. The parameter space is $\mathsf{Param} = \mathbb{R}^{m \times h} \times \mathbb{R}^{h \times n}$. Define the non-degenerate locus as:

$$(\mathsf{Param} \times \mathbb{R}^n)^\circ = \{(U, V, x) \in \mathbb{R}^{m \times h} \times \mathbb{R}^{h \times n} \times \mathbb{R}^n \mid Vx \neq 0\}$$

Let $\tilde{F} : \mathsf{Param} \times \mathbb{R}^n \to \mathbb{R}^m$ be the extended feedforward function, taking (U, V, x) to $F_{(U,V)}(x) = U\sigma(Vx)$. We now state and prove a more general version of Theorem 2.4.1.

Theorem A.3.4. *Suppose $\sigma(z) \neq 0$ for all* nonzero *$z \in \mathbb{R}^h \setminus \{0\}$.*

1. *There is an action:*

$$\mathrm{GL}_h \times (\mathsf{Param} \times \mathbb{R}^n)^\circ \to (\mathsf{Param} \times \mathbb{R}^n)^\circ$$

$$g \cdot (U,V,x) = (UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)} \ , \ gV \ , \ x)$$

2. *Suppose, in addition, that $\sigma(0) \neq 0$, so that σ is nonzero on all of $\mathbb{R}^h$. There is an action:*

$$\mathrm{GL}_h \times (\mathsf{Param} \times \mathbb{R}^n) \to (\mathsf{Param} \times \mathbb{R}^n)$$
$$g \cdot (U,V,x) = (UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)} \ , \ gV \ , \ x)$$

In both cases, the extended feedforward function is invariant for this action, that is: $\tilde{F}(g \cdot (U,V,x)) = \tilde{F}(U,V,x)$.

Proof. We first verify that the action is well-defined. In the second case, $\sigma(gVx) \neq 0$ for all (U,V,x) and hence $R_{\sigma(gVx)}$ is defined and invertible for any $g \in \mathrm{GL}_h$. For the first case, let (U,V,x) be in the non-degenerate locus. The non-degeneracy condition $Vx \neq 0$ guarantees that $gVx \neq 0$ for all $g \in \mathrm{GL}_h$. The hypothesis on σ in turn implies that $R_{\sigma(gVx)}$ is defined and invertible for any $g \in \mathrm{GL}_h$. Hence the action is well-defined in both cases.

To check the unit axiom, observe that, when $g = \mathrm{id}_h$ is the identity of GL_h, we have $R_{\sigma(Vx)}R^{-1}_{\sigma(gVx)} = R_{\sigma(Vx)}R^{-1}_{\sigma(Vx)} = \mathrm{id}_h$ and $gV = V$. It follows that $\mathrm{id} \cdot (U,V,x) = (U,V,x)$. To check the multiplication axiom, let $g_1, g_2 \in \mathrm{GL}_h$. Then:

$$\left(R_{\sigma(g_1g_2v)}R^{-1}_{\sigma(g_2v)}\right)\left(R_{\sigma(g_2v)}R^{-1}_{\sigma(v)}\right) = R_{\sigma(g_1g_2v)}R^{-1}_{\sigma(v)}.$$

It follows that $g_1 \cdot (g_2 \cdot (U,V,x)) = (g_1g_2) \cdot (U,V,x)$. For the last claim, we compute:

$$\begin{aligned}\tilde{F}(g \cdot (U,V,x)) &= \tilde{F}(UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)}, gV, x) = UR_{\sigma(Vx)}R^{-1}_{\sigma(gVx)}\sigma(gVx) \\ &= UR_{\sigma(Vx)}e_1 = U\sigma(Vx)\end{aligned}$$

where the first equality follows from the definition of the action; the second from the extended feedforward function $\tilde{F}$; and the third and fourth follow from Proposition A.3.3. □

From the proof, we see that a key property of the matrices R_z is that:

$$R_{\sigma(gz)}R_{\sigma(z)}^{-1}\sigma(z) = \sigma(gz) \tag{A.17}$$

This can be interpreted as a data-dependent generalization of the equivariance condition appearing in Equation equation 2.4. We emphasize that a sufficient condition for the existence of such an action is that $\sigma(z)$ is nonzero for any nonzero $z \in \mathbb{R}^h$; this is the case for usual sigmoid, hyperbolic tangent, leaky ReLU, and many other activations.

Finally, we remark on a differential-geometric interpretation of the construction of this section. One can regard σ as a section of the trivial bundle on $\mathbb{R}^h \setminus \{0\}$ with fiber $\mathbb{R}^h$. The map $z \mapsto R_{\sigma(gz)}R_{\sigma(z)}^{-1}$ defines a GL_h-equivariant structure on this bundle such that σ is an equivariant section. Indeed, the action of GL_h on the total space $\left(\mathbb{R}^h \setminus \{0\}\right) \times \mathbb{R}^h$ is given by $g \cdot (z,a) = (gz, R_{\sigma(gz)}R_{\sigma(z)}^{-1}a)$, and the equivariance of σ is precisely the condition $R_{\sigma(gz)}R_{\sigma(z)}^{-1}\sigma(z) = \sigma(gz)$.

A.3.3 Non-linear action: multi-layer case

We adopt the notation of Section A.2.2. In particular, consider a neural network with L layers and widths $\mathbf{n} = (n_0, n_1, \ldots, n_L)$. The parameter space is given by:

$$\mathsf{Param}(\mathbf{n}) = \mathbb{R}^{n_L \times n_{L-1}} \times \mathbb{R}^{n_{L-1} \times n_{L-2}} \times \cdots \times \mathbb{R}^{n_2 \times n_1} \times \mathbb{R}^{n_1 \times n_0} \times \mathbb{R}^{n_L} \times \mathbb{R}^{n_{L-1}} \times \cdots \times \mathbb{R}^{n_1}.$$

So for each layer i, we have a matrix $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and vector $b_i \in \mathbb{R}^{n_i}$. We write $\boldsymbol{\theta} = (W_i, b_i)_{i=1}^L$ for a choice of parameters. Fix activations $\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ for each $i = 1, \ldots, L$. Let

$$F = F_{\boldsymbol{\theta}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$$

be the feedforward function corresponding to parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}) \in \mathsf{Param}$ with activations σ_i. Taking the parameters into account, we form the extended feedforward function:

$$\widetilde{F} : \mathsf{Param} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}, \qquad \widetilde{F}(\boldsymbol{\theta}, x) = F_{\boldsymbol{\theta}}(x)$$

One can also define the extension of the partial feedforward function $\widetilde{F_i} : \mathsf{Param} \times \mathbb{R}^n \to \mathbb{R}^{n_i}$ as $(\boldsymbol{\theta}, x) \mapsto F_{\boldsymbol{\theta},i}(x)$. Furthermore, let $Z_i : \mathsf{Param} \times \mathbb{R}^n \to \mathbb{R}^{n_i}$ be the function defined recursively as:

$$Z_i(\boldsymbol{\theta}, x) = \begin{cases} x & \text{if } i = 0 \\ W_1 x + b_1 & \text{if } i = 1 \\ W_i \sigma_{i-1}(Z_{i-1}(\boldsymbol{\theta}, x)) + b_i & \text{for } i = 2, \ldots, L \end{cases}$$

We have $\tilde{F}_i = \sigma_i \circ Z_i$ for $i = 1, \ldots, L$, and the extended feedforward function is $\tilde{F} = \sigma_L \circ Z_L$. Define the non-degenerate locus as:

$$(\mathsf{Param} \times \mathbb{R}^n)^\circ = \{(\boldsymbol{\theta}, x) \mid Z_i(\boldsymbol{\theta}, x) \neq 0 \text{ for } i = 1, \ldots, L-1\}.$$

Proposition A.3.5. *Suppose that, for $i = 1, \ldots, L-1$, the activation $\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ satisfies $\sigma_i^{-1}(0) \subseteq \{0\}$. Then there is an action of the hidden symmetry group $\mathrm{GL}_{\mathbf{n}^{\text{hidden}}}$ on the non-degenerate locus given by:*

$$\mathrm{GL}_{\mathbf{n}^{\text{hidden}}} \times (\mathsf{Param} \times \mathbb{R}^n)^\circ \to (\mathsf{Param} \times \mathbb{R}^n)^\circ$$
$$g \cdot (\boldsymbol{\theta}, x) = \left(\left(g_i W_i R_{\tilde{F}_{i-1}(\boldsymbol{\theta}, x)} R^{-1}_{\sigma_i(g_{i-1} Z_{i-1}(\boldsymbol{\theta}, x))}, \; g_i b_i \right)_{i=1}^{L-1}, \; x \right)$$

Moreover, this action preserves the extended feedforward function.

Proof. The fact that the action is well-defined follows from the assumption on each σ_i and the non-degeneracy condition. The unit and multiplication axioms are shown in the same

way as in the proof of Theorem A.3.4. For the last claim, one first verifies by induction that $Z_i(g \cdot (\boldsymbol{\theta}, x)) = g_i Z_i(\boldsymbol{\theta}, x)$ for $i = 0, 1, \ldots, L$. Hence,

$$\tilde{F}(g \cdot \boldsymbol{\theta}, x) = \sigma_L(Z_L(g \cdot (\boldsymbol{\theta}, x))) = \sigma_L(g_L Z_L(\boldsymbol{\theta}, x)) = \sigma_L(Z_L(\boldsymbol{\theta}, x)) = \tilde{F}(\boldsymbol{\theta}, x)$$

using the fact that g_L is the identity. So the extended feedforward function is preserved under this action. □

A.3.4 Discussion: increasing the batch size

In this section, we discuss difficulties in adopting the construction of the previous sections to cases where the batch size is greater than one. Fix a batch size k, so that the feature space of the hidden layer is $\mathbb{R}^{h \times k}$. By abuse notation, we write $\sigma : \mathbb{R}^{h \times k} \to \mathbb{R}^{h \times k}$ for the map applying σ column-wise. We say that σ *preserves full rank matrices* if $\sigma(Z)$ is full rank for any full-rank matrix in $Z \in \mathbb{R}^{h \times k}$. As a final piece of notation, let $\left(\mathbb{R}^{h \times k}\right)^{\circ} \subseteq \mathbb{R}^{h \times k}$ be the subset of full rank matrices.

Lemma A.3.6. *Suppose that $k \leq h$, and that σ preserves full-rank matrices. Then there exists a map $c : \mathrm{GL}_h \times \left(\mathbb{R}^{h \times k} \setminus \{0\}\right) \to \mathrm{GL}_h$ satisfying the following identities for any nonzero $Z \in \mathbb{R}^{h \times k}$ and $g, g_1, g_2 \in \mathrm{GL}_h$:*

$$c(\mathrm{id}_h, Z) = \mathrm{id}_h \tag{A.18}$$

$$c(g_1, g_2 Z) c(g_2, Z) = c(g_1 g_2, Z) \tag{A.19}$$

$$c(g, Z)\sigma(Z) = \sigma(gZ) \tag{A.20}$$

We omit a proof of this lemma. A key tool is the fact that, for $k \leq h$, any two matrices in $\left(\mathbb{R}^{h \times k}\right)^{\circ}$ are related by an element of GL_h. This lemma implies that, for a multi-layer network, if σ_i preserves full rank matrices in $\mathbb{R}^{n_i \times k}$ for each i, then there is a non-linear group action as in

Proposition A.3.5, where the appropriate version of the non-degenerate locus is:

$$\Big(\mathsf{Param}\times\mathbb{R}^{n_0\times k}\Big)^{\circ}=\{(\boldsymbol{\theta},X)\mid Z_i(\boldsymbol{\theta},X)\in\mathbb{R}^{n_i\times k}\text{ is of full rank for }i=1,\ldots,L-1\}.$$

However, as the following examples show, the condition that σ preserves full rank matrices is not satisfied in the case of common activation functions.

Example A.3.7. *1. For $k>1$, the column-wise application of the usual sigmoid activation does not preserve full rank matrices. For example, for $k=2$, take:*

$$Z=\begin{bmatrix}\sigma^{-1}\left(\frac{1}{5}\right) & \sigma^{-1}\left(\frac{2}{5}\right)\\ \sigma^{-1}\left(\frac{2}{5}\right) & \sigma^{-1}\left(\frac{4}{5}\right)\end{bmatrix}\simeq\begin{bmatrix}-1.3863 & -0.4055\\ -0.4055 & 1.3863\end{bmatrix}$$

Then $\det(\sigma(Z))=0$ *while* $\det(Z)=-2.0862$.

2. *For $k>1$, the column-wise application of hyperbolic tangent does not preserve full rank matrices. To see this, set $k=1$ and consider:*

$$Z=\begin{bmatrix}\tanh^{-1}\left(\frac{1}{5}\right) & \tanh^{-1}\left(\frac{2}{5}\right)\\ \tanh^{-1}\left(\frac{2}{5}\right) & \tanh^{-1}\left(\frac{4}{5}\right)\end{bmatrix}\simeq\begin{bmatrix}0.2027 & 0.4236\\ 0.4236 & 1.0986\end{bmatrix}$$

Then $\det(\tanh(Z))=0$ *while* $\det(Z)=0.0432$.

3. *Let s be a real number with $0<s<1$. The corresponding leaky ReLU activation function is given by $\sigma(z)=sz\min(0,z)+z\max(0,z)$. For $k>1$, the column-wise application of leaky ReLU tangent does not preserve full rank matrices. Indeed, for $k=2$, set:*

$$Z=\begin{bmatrix}s & -1\\ -1 & s\end{bmatrix}$$

$$\textit{Then } \det(\sigma(Z)) = \det\left(\begin{bmatrix} s & -s \\ -s & s \end{bmatrix}\right) = 0 \textit{ while } \det(Z) = s^2 - 1 \neq 0.$$

Finally, in the case $k > h$, the action of GL_h on full rank $h \times k$ matrices is not transitive. Hence, there will generally be no matrix in GL_h taking $\sigma(Z)$ to $\sigma(gZ)$, even if both are full rank.

A.3.5 Lipschtiz bounds

Proof of Proposition 2.4.2. Let (U,V,x) be in the non-degenerate locus, let $g \in \mathrm{GL}_h$, and let $x_1, x_2 \in \mathbb{R}^n$. Using the Lipschitz constant of σ and the definition of operator norms, we compute:

$$\begin{aligned}
|F^{(g,x)}_{(U,V)}(x_1 - x_2)| &\leq |U R_{\sigma(Vx)} R^{-1}_{\sigma(gVx)} \sigma(gV(x_1 - x_2))| \\
&\leq \eta \|U\| \|R_{\sigma(Vx)}\| \|R^{-1}_{\sigma(gVx)}\| \|g\| \|V\| |x_1 - x_2| \\
&= \eta \|U\| |\sigma(Vx)| \| \frac{1}{|\sigma(gVx)|^2} R^T_{\sigma(gVx)} \| \|g\| \|V\| |x_1 - x_2| \\
&= \eta \|U\| \frac{|\sigma(Vx)| \|g\|}{|\sigma(gVx)|} \|V\| |x_1 - x_2|
\end{aligned}$$

The result follows. □

A.4 Drifting of Conserved Quantities under Gradient Descent

While gradient flows are well approximated by gradient descent [40], the conserved quantities of gradient descent are no longer conserved in gradient flow due to non-infinitesimal time steps. However, with small learning rate, we expect the change in the conserved quantities to be small. In this section, we first prove that the change of Q is bounded by the square of learning rate for two layer linear networks, then show empirically that the change Q is small for nonlinear networks.

A.4.1 Change in Q in gradient descent (linear layers)

Proposition A.4.1. *Consider the two layer linear network, where $U \in \mathbb{R}^{m\times h}, V \in \mathbb{R}^{h\times n}$ are the only parameters, and the loss function L is a function of UV. In gradient descent with learning rate η, the change in the conserved quantity $Q = \mathrm{Tr}\left[U^T U - VV^T\right]$ at step t is bounded by*

$$|Q_{t+1} - Q_t| \leq \eta^2 \left|\frac{dL(t)}{dt}\right|. \tag{A.21}$$

Proof. Let U_t and V_t be the value of U and V at time t in a gradient descent. The update rule is

$$U_{t+1} = U_t - \eta \frac{\partial L}{\partial U}, \qquad V_{t+1} = V_t - \eta \frac{\partial L}{\partial V} \tag{A.22}$$

Consider the two layer linear reparametrization $W = UV$.

$$\begin{aligned} Q_t &= \mathrm{Tr}\left[U_t^T U_t - V_t V_t^T\right] \\ Q_{t+1} &= \mathrm{Tr}\left[U_{t+1}^T U_{t+1} - V_{t+1} V_{t+1}^T\right] \end{aligned} \tag{A.23}$$

Substituting in U_{t+1} and V_{t+1}, expanding Q_{t+1}, and subtracting by Q_t, we have

$$\begin{aligned} Q_{t+1} - Q_t = \mathrm{Tr}\Bigg[&\eta^2 \left(\frac{\partial L}{\partial U_t}\right)^T \frac{\partial L}{\partial U_t} - \eta \left(\frac{\partial L}{\partial U_t}\right)^T U_t - \eta U_t^T \frac{\partial L}{\partial U_t} \\ &- \eta^2 \frac{\partial L}{\partial V_t}\left(\frac{\partial L}{\partial V_t}\right)^T + \eta \frac{\partial L}{\partial V_t} V_t^T + \eta V_t \left(\frac{\partial L}{\partial V_t}\right)^T \Bigg]. \end{aligned} \tag{A.24}$$

Note that

$$\left(\frac{\partial L}{\partial U_t}\right)^T U_t = (\nabla L V_t^T)^T U_t = V_t \nabla L^T U_t = V_t \left(\frac{\partial L}{\partial V_t}\right)^T, \tag{A.25}$$

and similarly

$$U_t^T \frac{\partial L}{\partial U_t} = \frac{\partial L}{\partial V_t} V_t^T. \tag{A.26}$$

Therefore, equation A.24 simplifies to

$$\begin{aligned} Q_{t+1} - Q_t &= \eta^2 \operatorname{Tr}\left[\left(\frac{\partial L}{\partial U_t}\right)^T \frac{\partial L}{\partial U_t} - \frac{\partial L}{\partial V_t}\left(\frac{\partial L}{\partial V_t}\right)^T\right] \\ &= \eta^2 \left(\operatorname{Tr}\left[\left(\frac{\partial L}{\partial U_t}\right)^T \frac{\partial L}{\partial U_t}\right] - \operatorname{Tr}\left[\left(\frac{\partial L}{\partial V_t}\right)^T \frac{\partial L}{\partial V_t}\right]\right), \end{aligned} \tag{A.27}$$

and the variation of Q in each step is bounded by the convergence rate:

$$\begin{aligned} |Q_{t+1} - Q_t| &= \eta^2 \left|\operatorname{Tr}\left[\left(\frac{\partial L}{\partial U_t}\right)^T \frac{\partial L}{\partial U_t}\right] - \operatorname{Tr}\left[\left(\frac{\partial L}{\partial V_t}\right)^T \frac{\partial L}{\partial V_t}\right]\right| \\ &\leq \eta^2 \left|\operatorname{Tr}\left[\left(\frac{\partial L}{\partial U_t}\right)^T \frac{\partial L}{\partial U_t}\right] + \operatorname{Tr}\left[\left(\frac{\partial L}{\partial V_t}\right)^T \frac{\partial L}{\partial V_t}\right]\right| \\ &= \eta^2 \left|\frac{dL}{dt}\right| \end{aligned} \tag{A.28}$$

□

A.4.2 Empirical observations

In gradient flow, the conserved quantity Q is constant by definition. In gradient descent, Q varies with time. In order to see how applicable our theoretical results are in gradient descent, we investigate the amount of variation in Q in gradient descent using two-layer neural networks.

Since Q is the difference between the two terms $f_1(U) = \frac{1}{2}\operatorname{Tr}[U^T U]$ and $f_2(V) = \sum_{a,j} \int_{x_0}^{V_{aj}} dx \frac{\sigma(x)}{\sigma'(x)}$, we normalize Q by the initial value of $f_1(U)$ and $f_2(V)$, i.e.,

$$\tilde{Q} = \frac{\left|\frac{1}{2}\operatorname{Tr}[U^T U] - \sum_{a,j} \int_{x_0}^{V_{aj}} dx \frac{\sigma(x)}{\sigma'(x)}\right|}{\left|\frac{1}{2}\operatorname{Tr}[U_0^T U_0]\right| + \left|\sum_{a,j} \int_{x_0}^{V_{0aj}} dx \frac{\sigma(x)}{\sigma'(x)}\right|}$$

and denote the amount of change in $\tilde{Q}$ as

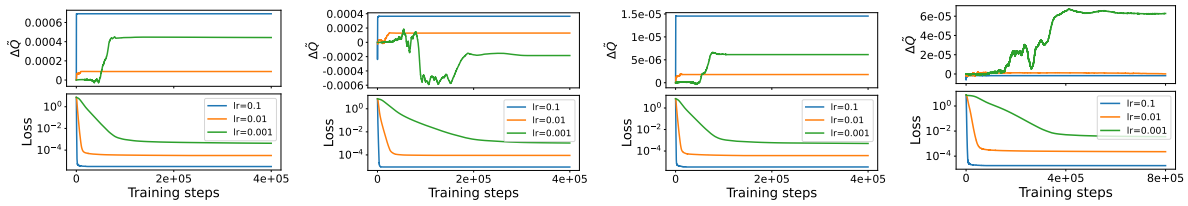
$$\Delta\tilde{Q}(t) = \tilde{Q}(t) - \tilde{Q}(0) \tag{A.29}$$

We run gradient descent on two-layer networks with whitened input with the following objective

$$\text{argmin}_{U,V}\{L(U,V) = \|Y - U\sigma(V^T)\|_F^2\} \tag{A.30}$$

where σ is the identity function, ReLU, sigmoid, or tanh. $Y \in \mathbb{R}^{5\times 10}$, $U \in \mathbb{R}^{5\times 50}$ and $V \in \mathbb{R}^{10\times 50}$ have random Gaussian initialization with zero mean. We repeat the gradient descent with learning rate 0.1, 0.01, and 0.001.

The variation $\Delta\tilde{Q}(t)$ and loss is shown in Fig.A.1. The amount of change in Q is small relative to the magnitude of $f_1(U)$ and $f_2(V)$, indicating that conserved quantities in gradient flow are approximately conserved in gradient descent. The error in Q grows with step size, as $\Delta\tilde{Q}(t)$ is larger with the largest learning rate we used, although it has the same magnitude as those of smaller learning rates. We also observe that Q stays constant after loss converges.



(a) Linear. (b) ReLU. (c) Tanh. (d) Sigmoid.

Figure A.1. Dynamics of conserved quantities in GD. The amount of change in Q is small relative to its magnitude, and Q converges when loss converges.

Appendix B

Supplementary Material for Chapter 3

B.1 Omitted Proofs in Chapter 3.2

Proposition 3.2.1. *There is a homeomorphism between* $L^{-1}(0)$ *and* $(\mathrm{GL}_h)^{l-1}$.

Proof. Recall that $W_1, ..., W_n, X, Y$ are matrices in $\mathbb{R}^{h\times h}$, and X, Y are both full rank. Consider the map

$$f : (\mathrm{GL}_h)^{l-1} \to L^{-1}(0), \quad (g_1, ..., g_{l-1}) \mapsto (g_1 X^{-1}, g_2, ..., g_{l-1}, Y\prod_{i}^{l-1} g_i^{-1}). \tag{B.1}$$

The inverse $f^{-1} : (W_1, ..., W_l) \mapsto (W_1 X, W_2, W_3, ..., W_{l-1})$ is well defined, because X, W_1, W_2, W_3, $..., W_{l-1}$ are all full-rank. Since both f and f^{-1} are continuous, f is a homeomorphism between $(\mathrm{GL}_h)^{l-1}$ and $L^{-1}(0)$. □

Corollary 3.2.2. *The minimum of L has* 2^{l-1} *connected components.*

Proof. From Proposition 3.2.1, $L^{-1}(0)$ is homeomorphic to $(\mathrm{GL}_h)^{l-1}$. This implies that $L^{-1}(0)$ has the same number of connected components as $(\mathrm{GL}_h)^{l-1}$. Therefore, $GL_h(\mathbb{R})^{l-1}$ has 2^{l-1} connected components. Therefore, $L^{-1}(0)$ has 2^{l-1} connected components. □

Proposition 3.2.3. *Let* $n = 1$. *Assume that* $X, Y \neq 0$. *When* $\varepsilon = 0$, *the minimum of L has 4 connected components. When* $\varepsilon \neq 0$, *the minimum of L has 3 connected components.*

Proof. When $\varepsilon = 0$, the skip connection is effectively removed, and the loss function equation 3.2 reduces to equation 3.1. By Corollary 3.2.2, the minimum of L has 4 connected components. In the rest of the proof, we consider the case where $\varepsilon \neq 0$.

Let $(W_{1_0}, W_{2_0}, W_{3_0}) = (I, (\alpha - \varepsilon)I, \alpha^{-1}YX^{-1})$, where $\alpha \in \mathbb{R}$ is an arbitrary number such that $\alpha \neq \varepsilon$ and $\alpha \neq 0$. Then $(W_{1_0}, W_{2_0}, W_{3_0})$ is a point in $L^{-1}(0)$. Define set $G_1 = \{g \in R^{h \times h} : \det(gW_{2_0}W_{1_0}X + \varepsilon X) \neq 0\}$. Let $a : GL_1 \times G_1 \to \mathsf{Param}$ be the following map:

$$\begin{aligned} g_1, g_2 \mapsto (&g_1 W_{1_0}, \\ &g_2 W_{2_0} g_1^{-1}, \\ &W_{3_0}(W_{2_0}W_{1_0}X + \varepsilon X)(g_2 W_{2_0}W_{1_0}X + \varepsilon X)^{-1}). \end{aligned} \tag{B.2}$$

From the definition of G_1, $(g_2W_{2_0}W_{1_0}X + \varepsilon X)$ is invertible, so a is well defined. Additionally, we have $L(a(g_1, g_2)) = L(W_{1_0}, W_{2_0}, W_{3_0}) = 0, \forall g_1, g_2 \in GL_1 \times G_1$. Therefore, denoting the image of a as S_1, we have $S_1 \subseteq L^{-1}(0)$.

Let $S_0 = \{(W_1, W_2, W_3) : W_3 = Y(\varepsilon X)^{-1} \text{ and } W_1 = 0\}$ if $\varepsilon \neq 0$, or $\emptyset$ otherwise. For $(W_1, W_2, W_3) \in S_0$, we have $L(W_1, W_2, W_3) = ||Y - Y(\varepsilon X)^{-1}(0 + \varepsilon X)||_2 = 0$. Therefore, $S_0 \subseteq L^{-1}(0)$.

We then show that the minimum of L is the union of S_1 and S_0. Consider a point $(W_1, W_2, W_3) \in L^{-1}(0)$. If $W_1 = 0$, then $\varepsilon \neq 0$, otherwise (W_1, W_2, W_3) cannot be in $L^{-1}(0)$. In this case, W_3 must equal to $Y(\varepsilon X)^{-1}$, and $(W_1, W_2, W_3) \in S_0$. If $W_1 \neq 0$, then $W_1 W_{1_0}^{-1} \in GL_1$ and $W_2 W_1 W_{1_0}^{-1} W_{2_0}^{-1} \in G_1$. The second part is due to $W_2 W_1 W_{1_0}^{-1} W_{2_0}^{-1} W_{2_0} W_{1_0} X + \varepsilon X = W_2 W_1 X + \varepsilon X \neq 0$ since $(W_1, W_2, W_3) \in L^{-1}(0)$. In this case we have $(W_1, W_2, W_3) = a(W_1 W_{1_0}^{-1}, W_2 W_1 W_{1_0}^{-1} W_{2_0}^{-1})$, which means that $(W_1, W_2, W_3) \in S_1$.

The number of connected components of S_1 and S_0 can be obtained from their structures. Since $W_{2_0}W_{1_0}X \neq 0$, there is a homeomorphism between G_1 and GL_1 defined by the map

$$f : G_1 \to GL_1, g \mapsto gW_{2_0}W_{1_0}X + \varepsilon X \tag{B.3}$$

with inverse $f^{-1}: GL_1 \to G_1, g \mapsto \varepsilon(g-\varepsilon X)(W_{2_0}W_{1_0}X)^{-1}$. Since a is also a homeomorphism, its image S_1 is homeomorphic to $GL_1 \times GL_1$ and has 4 connected components. When $\varepsilon \neq 0$, S_0 is a line and thus has 1 connected component.

The last part of the proof shows the connectedness of the connected components of S_1 and S_0. Let $G_1^+ = \{g_2 \in G_1 : f(g_2) \in GL^{sign(\varepsilon X)}\}$ be the connected component in G_1 that correspond to $GL^{sign(\varepsilon X)}$, and $G_1^- = \{g_2 \in G_1 : f(g_2) \in GL^{-sign(\varepsilon X)}\}$ be the component that correspond to $GL^{-sign(\varepsilon X)}$. For convenience, we name the connected components of $Im(a)$ as follows:

$$C_1 = \{(W_1,W_2,W_3) \in \mathsf{Param} : (W_1,W_2,W_3) = a(g_1,g_2), g_1 \in GL^+, g_2 \in G_1^+\}$$
$$C_2 = \{(W_1,W_2,W_3) \in \mathsf{Param} : (W_1,W_2,W_3) = a(g_1,g_2), g_1 \in GL^-, g_2 \in G_1^+\}$$
$$C_3 = \{(W_1,W_2,W_3) \in \mathsf{Param} : (W_1,W_2,W_3) = a(g_1,g_2), g_1 \in GL^+, g_2 \in G_1^-\}$$
$$C_4 = \{(W_1,W_2,W_3) \in \mathsf{Param} : (W_1,W_2,W_3) = a(g_1,g_2), g_1 \in GL^-, g_2 \in G_1^-\}$$

Note that for $(W_1,W_2,W_3) \in S_1$, there exists a (unique) $g_2 \in G_1$ such that we can write W_3 as

$$W_3 = W_{3_0}[W_{2_0}W_{1_0}X + \varepsilon X][g_2 W_{2_0}W_{1_0}X + \varepsilon X]^{-1}) = Y f(g_2)^{-1}.$$

Following the definition of G_1^+, for a point (W_1,W_2,W_3) in C_1 or C_2, $sign(W_3) = sign(Y(\varepsilon X)^{-1})$. Additionally, when g_2 is close to 0, g_2 belongs to G_1^+. The boundary of both C_1 and C_2 contain a point in S_0:

$$\lim_{g_1 \to 0^+} a(g_1,g_1) = \lim_{g_1 \to 0^-} a(g_1,g_1) = (0, \alpha - \varepsilon, Y(\varepsilon X)^{-1}) \in S_0.$$

Therefore, both C_1 and C_2 are connected to S_0.

For points in C_3 and C_4, $sign(W_3) \neq sign(Y(\varepsilon X)^{-1})$. Therefore, no point in C_3 or C_4 can be sufficiently close to S_0. As a result, these components are not connected to S_0. In

summary, when $\varepsilon \neq 0$, S_0 connects 2 components of S_1, and the minimum of L has 3 connected components. □

We note that connectedness alone does not imply easy connectivity in the sense of short or simple paths between solutions. Being in the same connected components is a necessary condition for connectivity, but a single component may still contain complex geometry necessitating complicated connecting paths.

Defining the ease of connectivity is subtle. One natural measure is the parametric complexity of the connecting curves, quantifiable by their degree if polynomial, or number of segments if piece-wise. Another possible definition for easy connectivity would be low curvature of the minimum manifold or short geodesic distance between two points on it. As we saw in Section 3.4.2, low curvature implies that linear interpolation stays near the manifold. Other potential definitions include whether the connecting curve has an analytical expression, or how many points are needed to approximate it within a certain error. It would be interesting to examine these properties for symmetry-induced connecting curves.

B.2 Omitted Proofs in Chapter 3.3

Lemma 3.3.1. *Consider two points* $(W_1, W_2), (W_1', W_2') \in L^{-1}(0)$ *that are not connected in* $L^{-1}(0)$. *For any* $g \in GL(h)$ *such that* $det(g) < 0$, $g \cdot (W_1, W_2)$ *and* (W_1', W_2') *are connected in* $L^{-1}(0)$.

Proof. Consider the map f and its inverse f^{-1} defined in equation B.1 in the proof of Proposition 3.2.1. Let $g = f^{-1}(W_1, W_2)$ and $g' = f^{-1}(W_1', W_2')$. Since (W_1, W_2) and (W_1', W_2') are not in the same connected component of $L^{-1}(0)$, g and g' are not in the same connected component of GL_h. Equivalently, $det(gg') < 0$. Consider a $g_1 \in GL_h$ such that $det(g) < 0$. Then $det(g_1 g g') > 0$, which means that $g_1 g$ and g' belong to the same connected component of GL_h. Therefore, $g_1 \cdot (W_1, W_2) = f(g_1 g)$ and $(W_1', W_2') = f(g')$ belong to the same connected component of $L^{-1}(0)$. □

Example.

Suppose $\left(W_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, W_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\right)$ is a point in $L^{-1}(0)$ for some loss function L. Then $\left(W_1' = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, W_2' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$ is also a point in $L^{-1}(0)$. However, (W_1, W_2) and (W_1', W_2') are not on the same connected component of the minimum, since their determinants have different signs. By Lemma 3.3.1, any $g \in GL(h)$ with $det(g) < 0$ can bring (W_1, W_2) and (W_1', W_2') to the same connected component in $L^{-1}(0)$. Let g be the permutation matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Then $g \cdot (W_1, W_2) = \left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right)$, which is in the same connected component as (W_1', W_2').

Proposition 3.3.2. *Assume that $h \geq 2$. For all $(W_1, ..., W_l), (W_1', ..., W_l') \in L^{-1}(0)$, these exists a list of permutation matrices $P_1, ..., P_{l-1}$ such that $(W_1P_1, P_1^{-1}W_2P_2, ..., P_{l-2}W_{l-1}P_{l-1}, P_{l-1}W_l)$ and $(W_1', ..., W_l')$ are connected in $L^{-1}(0)$.*

Proof. Let $(g_1, ..., g_{l-1}), (g_1', ..., g_{l-1}') \in (GL_h)^{n-1}$ such that $f(g_1, ..., g_{l-1}) = (W_1, ..., W_l)$ and $f(g_1', ..., g_{l-1}') = (W_1', ..., W_l')$. Let $P_0 = I$. For $i = 1, ..., l-1$, if $det(g_i g_i' P_{i-1}^{-1}) > 0$, set P_i to I. Otherwise, we set P_i to an arbitrary element in $P \in S_h \setminus A_h$, which is not empty when $h \geq 2$.

Let $(g_1'', ..., g_{l-1}'') \in (GL_h)^{n-1}$ s.t. $f(g_1'', ..., g_{l-1}'') = (W_1P_1, P_1^{-1}W_2P_2, ..., P_{l-2}W_{l-1}P_{l-1}, P_{l-1}W_l)$. By the way we construct P_i's, we have $g_i'' = P_{i-1}^{-1} g_i' P_i$ and $det(g_i g_i'') > 0$. Therefore, g_i and g_i'' belong to the same connected component of $(GL_h)^{l-1}$ for all i. Since f is a homeomorphism between $(\mathrm{GL}_h)^{l-1}$ and $L^{-1}(0)$, $(W_1P_1, P^{-1}W_2P_2, ..., P_{l-2}W_{l-1}P_{l-1}, P_{l-1}W_l)$ and $(W_1', ..., W_l')$ are connected in $L^{-1}(0)$. □

Proposition 3.3.3. *Consider the loss function of the following form*

$$L : \mathsf{Param} \to \mathbb{R}, W = (W_1, ..., W_l) \mapsto ||Y - W_l \sigma(W_{l-1} f(W_{l-2}, W_{l-3}, ..., W_1, X))||_2^2, \qquad \text{(B.4)}$$

where f is a function of $W_{l-2}, W_{l-3}, ..., W_1, X$, and $\sigma(cz) = c^k \sigma(z)$ for all $c \in \mathbb{R}$ and some $k > 0$.

Assume that $||Y||_2 \neq 0$ *and* $L^{-1}(0) \neq \emptyset$*. Also assume that* $l \geq 2$*. For any positive number* $b > 0$*, there exist* $W, W' \in L^{-1}(0)$ *that belong to the same connected component of* $L^{-1}(0)$ *and* $0 < \alpha < 1$*, such that* $L((1-\alpha)W + \alpha W') > b$*.*

Proof. Let $W = (W_l, ..., W_2, W_1) \in L^{-1}(0)$ be an arbitrary point on the minimum of L. Let $W' = (W_l', ..., W_2', W_1') = (W_l m^{-k}, mW_{l-1}, W_{l-2}, ..., W_1)$. Then W, W' belong to the same connected component of $L^{-1}(0)$, connected by curve $\gamma : \mathbb{R} \to \mathsf{Param}, \gamma(t) = ((1-t)W_l + tW_l m^{-k}, (1-t)W_{l-1} + tmW_{l-1}, W_{l-2}, ..., W_1)$.

Since $W \in L^{-1}(0)$, we have $W_l \sigma[W_{l-1} f(W_{l-2}, ..., W_1, X)] = Y$. The loss on the linear interpolation of W, W' is

$$\begin{aligned} L\big((1-\alpha)W + \alpha W'\big) =& ||Y - ((1-\alpha)W_l + \alpha W_l')\sigma\big[((1-\alpha)W_{l-1} + \alpha W_{l-1}') \\ & f(W_{l-2}, ..., W_1, X)\big]||_2^2 \\ =& ||Y - (1-\alpha+\alpha m^{-k})W_l \sigma[(1-\alpha+\alpha m)W_{l-1} f(W_{l-2}, ..., W_1, X)]||_2^2 \\ =& ||Y - (1-\alpha+\alpha m^{-k})(1-\alpha+\alpha m)^k W_l \sigma[W_{l-1} f(W_{l-2}, ..., W_1, X)]||_2^2 \\ =& (1 - (1-\alpha+\alpha m^{-k})(1-\alpha+\alpha m)^k)^2 ||Y||_2^2. \end{aligned} \tag{B.5}$$

Let $\alpha = 0.5$. Then

$$\begin{aligned} L\big((1-\alpha)W + \alpha W'\big) &= \left(1 - \left(\frac{1}{2} + \frac{1}{2}m^{-k}\right)\left(\frac{1}{2} + \frac{1}{2}m\right)^k\right)^2 ||Y||_2^2 \\ &= \left(1 - 2^{-(k+1)}(1+m^{-k})(1+m)^k\right)^2 ||Y||_2^2 \end{aligned} \tag{B.6}$$

Let $m = \left(2^{k+1}\left(\frac{\sqrt{b}}{||Y||^2} + 1\right) - 1\right)^k$. Recall that $k > 0$. Then $m > 0$, $(1+m)^k > 1$, and

$$2^{-(k+1)}(1+m^{-k})(1+m)^k > 2^{-(k+1)}(1+m^{-k}) = \frac{\sqrt{b}}{||Y||^2} + 1 > 1. \tag{B.7}$$

Therefore, the loss at our chosen values of α and m is at least b:

$$L\left((1-\alpha)W+\alpha W'\right) > \left(1-\left(\frac{\sqrt{b}}{||Y||^2}+1\right)\right)^2 ||Y||_2^2 = b. \tag{B.8}$$

□

Proposition 3.3.4. *Consider the loss function with the same set of assumptions in Proposition 3.3.3. Assume additionally that there does not exist a permutation P such that every column of* $P\sigma(W_{l-1}f(W_{l-2},W_{l-3},...,W_1,X))$ *is in the null space of* W_l*. For any positive number* $b>0$*, there exist* $(W_1,...,W_l),(W_1',...,W_l') \in L^{-1}(0)$ *and* $0<\alpha<1$*, such that* $(W_1,...,W_{l-2})=(W_1',...,W_{l-2}')$ *and* $\min_{P\in S_n} L\left((1-\alpha)(W_1,...,W_l)+\alpha(W_1,...,W_{l-2},P^{-1}W_{l-1},W_lP)\right) > b$*.*

Proof. Let $W=(W_l,...,W_2,W_1)\in L^{-1}(0)$ be an arbitrary point on the minimum of L. Let $W'=(W_l',...,W_2',W_1')=(W_lm^{-k},mW_{l-1},W_{l-2},...,W_1)$.

Since $W\in L^{-1}(0)$, we have $W_l\sigma\left[W_{l-1}f(W_{l-2},...,W_1,X)\right]=Y$. The loss on the linear interpolation of W,W' is

$$\begin{aligned} &L\left((1-\alpha)W+\alpha W'\right) \\ &= ||Y-((1-\alpha)W_l+\alpha W_l'P)\sigma\left[((1-\alpha)W_{l-1}+\alpha P^{-1}W_{l-1}')f(W_{l-2},...,W_1,X)\right]||_2^2. \end{aligned} \tag{B.9}$$

Let $\alpha=0.5$. Then

$$L\left((1-\alpha)W+\alpha W'\right) = ||Y-\frac{1}{4}W_l(I+m^{-k}P)\sigma\left[(I+mP^{-1})W_{l-1}f(W_{l-2},...,W_1,X)\right]||_2^2. \tag{B.10}$$

When $m\to\infty$,

$$\begin{aligned} &\lim_{m\to\infty}\sigma\left[(I+mP^{-1})W_{l-1}f(W_{l-2},...,W_1,X)\right] \\ =&\lim_{m\to\infty} m^k\sigma\left[(m^{-1}I+P^{-1})W_{l-1}f(W_{l-2},...,W_1,X)\right] \end{aligned}$$

$$= \lim_{m\to\infty} m^k P^{-1} \sigma\left[W_{l-1} f(W_{l-2}, ..., W_1, X)\right]. \tag{B.11}$$

Therefore,

$$\begin{aligned}
\lim_{m\to\infty} L\left((1-\alpha)W + \alpha W'\right) &= \lim_{m\to\infty} ||Y - \frac{1}{4} W_l (I + m^{-k}P) m^k P^{-1} \sigma\left[W_{l-1} f(W_{l-2}, ..., W_1, X)\right]||_2^2 \\
&= \lim_{m\to\infty} ||Y - \frac{1}{4} W_l (I + m^k P^{-1}) \sigma\left[W_{l-1} f(W_{l-2}, ..., W_1, X)\right]||_2^2 \\
&= \lim_{m\to\infty} ||\frac{3}{4} Y - \frac{m^k}{4} W_l P^{-1} \sigma\left[W_{l-1} f(W_{l-2}, ..., W_1, X)\right]||_2^2.
\end{aligned} \tag{B.12}$$

Since we assumed that there does not exist a permutation P such that every column of $P\sigma(W_{l-1} f(W_{l-2}, W_{l-3}, ..., W_1, X))$ is in the null space of W_l, at least one element in the second term is unbounded for any permutation P. Therefore, $L\left((1-\alpha)W + \alpha W'\right)$ is unbounded for any P. □

Proposition 3.3.5. *Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix. Let set $S = \{(W_1, W_2) : W_1, W_2 \in \mathbb{R}^{n\times n}, W_1 W_2 = A\}$. For any positive number $b > 0$, there exist $W', W'' \in S$ and $0 < \alpha < 1$, such that $\min_{\hat{W}\in S} \| \left((1-\alpha)W' + \alpha W''\right) - \hat{W} \|_2 > b$.*

Proof. Let W be an element of S. Let $W_1' = W_1 g_1^{-1}, W_2' = g_1 W_2, W_1'' = W_1 g_2^{-1}$, and $W_2'' = g_2 W_2$, where $g_1, g_2 \in \mathbb{R}^{n\times n}$ are invertible matrices. Note that $W' = (W_1', W_2')$ and $W'' = (W_1'', W_2'')$ are both in S. Then,

$$\begin{aligned}
&\min_{\hat{W}\in S} \| \left((1-\alpha)W' + \alpha W''\right) - \hat{W} \|_2^2 \\
=&\min_{\hat{W}\in S} \|(1-\alpha) W_1 g_1^{-1} + \alpha W_1 g_2^{-1} - \hat{W}_1\|_2^2 + \|(1-\alpha) g_1 W_2 + \alpha g_2 W_2 - \hat{W}_2\|_2^2 \\
=&\min_{g\in GL(n)} \|W_1((1-\alpha) g_1^{-1} + \alpha g_2^{-1} - g^{-1})\|_2^2 + \|W_2((1-\alpha) g_1 + \alpha g_2 - g)\|_2^2.
\end{aligned} \tag{B.13}$$

Let $g_1 = \beta I$ and $g_2 = \beta^{-1} I$ for some $\beta > 0$. Let $\alpha = \frac{1}{2}$. Then, in the limit of a large β,

we have

$$\lim_{\beta\to\infty}\min_{\hat{W}\in S}\|\left((1-\alpha)W+\alpha W'\right)-\hat{W}\|_2^2$$
$$=\lim_{\beta\to\infty}\min_{g\in GL(n)}\left\|W_1\left(\frac{\beta+\beta^{-1}}{2}I-g^{-1}\right)\right\|_2^2+\left\|W_2\left(\frac{\beta+\beta^{-1}}{2}I-g\right)\right\|_2^2. \tag{B.14}$$

As $\beta\to\infty$, g and g^{-1} cannot approach $\frac{\beta+\beta^{-1}}{2}I$ simultaneously. Therefore, equation B.14 is not bounded. □

Proposition 3.3.6. *Consider the loss function with the same set of assumptions in Proposition 3.3.3. Let $W\in L^{-1}(0)$ be a point on the minimum. Consider the multiplicative group of positive real numbers $\mathbb{R}^+$ that acts on $L^{-1}(0)$ by $g\cdot(W_1,...,W_l)=(W_1,...,W_{l-2},gW_{l-1},W_lg^{-k})$, where $g\in\mathbb{R}^+$. Then there exists a positive number $b>0$, such that for all $0<\alpha<1$ and $W'\in Orbit(W)$ with $||W_i'||_2<c$ for all i and some $c>0$, the loss value for points on the linear interpolation $L((1-\alpha)W+\alpha W')<b$.*

Proof. Since $W'\in Orbit(W)$, $W'=(W_lm^{-k},mW_{l-1},W_{l-2},...,W_1)$ for some $m>0$. Additionally, m and m^{-k} are bounded since W_i' is bounded.

Since $W\in L^{-1}(0)$, we have $W_l\sigma[W_{l-1}f(W_{l-2},...,W_1,X)]=Y$. The loss on the linear interpolation of W,W' is

$$\begin{aligned}L\left((1-\alpha)W+\alpha W'\right)=&||Y-((1-\alpha)W_l+\alpha W_l')\\&\sigma\left[((1-\alpha)W_{l-1}+\alpha W_{l-1}')f(W_{l-2},...,W_1,X)\right]||_2^2\\=&||Y-(1-\alpha+\alpha m^{-k})W_l\sigma\left[(1-\alpha+\alpha m)W_{l-1}f(W_{l-2},...,W_1,X)\right]||_2^2\\=&||Y-(1-\alpha+\alpha m^{-k})(1-\alpha+\alpha m)^kW_l\sigma\left[W_{l-1}f(W_{l-2},...,W_1,X)\right]||_2^2\\=&(1-(1-\alpha+\alpha m^{-k})(1-\alpha+\alpha m)^k)^2||Y||_2^2.\end{aligned} \tag{B.15}$$

As m, m^{-k}, and α are all bounded, the loss value for points on the linear interpolation, $L((1-\alpha)W+\alpha W')$, is also bounded. □

The connectedness results derived from symmetry raise several interesting questions about mode connectivity. For example, it would be interesting to understand when and why there is no significant change in loss on the linear interpolation between two minima. One possible explanation is that there always exists a symmetry-induced path γ that stays close to the linear interpolation. Another potential factor is the high dimensionality of the minimum, which increases the likelihood that a significant portion of the linear interpolation remains within the low-loss region. Additionally, empirical observations suggest that both train and test accuracy remain nearly constant along paths connecting different SGD solutions [52]. If these paths are induced by a group action, this would imply that the group action's dependence on data is weak. Investigating the extent to which data influences these symmetries could provide deeper insights into the structure of the loss landscape and the generalization properties of neural networks.

B.3 Omitted Proofs in Chapter 3.4

Proposition 3.4.1. *Let* $(U,V) \in \mathsf{Param}$*, and* $(U',V') = g \cdot (U,V)$*. Then*

$$\|U\sigma(VX) - U'\sigma(V'X)\| \leq \|U\sigma(VX)\|. \tag{B.16}$$

Proof. We note that $I - \sigma(gVX)^\dagger \sigma(gVX)$ is a projection:

$$\begin{aligned}
&(I - \sigma(gVX)^\dagger \sigma(gVX))^2 \\
=&I - \sigma(gVX)^\dagger \sigma(gVX) - \sigma(gVX)^\dagger \sigma(gVX)(I - \sigma(gVX)^\dagger \sigma(gVX)) \\
=&I - \sigma(gVX)^\dagger \sigma(gVX).
\end{aligned}$$

Therefore,

$$\|U\sigma(VX) - U'\sigma(V'X)\| = \|U\sigma(VX)\left(I - \sigma(gVX)^\dagger \sigma(gVX)\right)\| \leq \|U\sigma(VX)\|. \tag{B.17}$$

□

Theorem 3.4.2. *Let $L^{-1}(c) \subset \mathsf{Param}$, with $c \in \mathbb{R}$, be a level set of the loss function $L : \mathsf{Param} \to \mathbb{R}$. Let $\gamma : [0,1] \to L^{-1}(c)$ be a smooth curve in $L^{-1}(c)$ connecting two points $\boldsymbol{w}_1 = \gamma(0)$ and $\boldsymbol{w}_2 = \gamma(1)$. Suppose the curvature $\kappa(t)$ of γ satisfies $\kappa(t) \leq \kappa_{\max}$ for all $t \in [0,1]$.*

Let S be the straight line segment connecting $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Then, for any point $\boldsymbol{w}$ on S, the distance to $L^{-1}(c)$ is bounded by

$$\operatorname{dist}(\boldsymbol{w}, L^{-1}(c)) \leq d_{\max} = \frac{1}{\kappa_{\max}} \left(1 - \sqrt{1 - \left(\frac{\kappa_{\max} \|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2}{2} \right)^2} \right).$$

Furthermore, assuming L is Lipschitz continuous with Lipschitz constant C_L, the loss at any point $\boldsymbol{w}$ on S satisfies

$$|L(\boldsymbol{w}) - c| \leq C_L d_{\max}.$$

Proof. We will find an upper bound for the maximum distance between a smooth curve and the chord connecting two points on the curve, assuming the curvature of the curve is bounded by $\kappa_{\max}$.

The curvature κ at a point on a curve is defined as $\kappa = \frac{1}{R}$, where R is the radius of the osculating circle at that point. Let s be the maximum perpendicular distance from the midpoint of a chord to the curve. For a circular arc, Pythagorean theorem gives

$$R^2 = \left(\frac{\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2}{2} \right)^2 + (R - s)^2.$$

Solving for s:

$$s = R \left(1 - \sqrt{1 - \left(\frac{\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2}{2R} \right)^2} \right).$$

Substitute $R = \frac{1}{\kappa}$ into the above, we have

$$s = \frac{1}{\kappa}\left(1 - \sqrt{1 - \left(\frac{\kappa\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2}{2}\right)^2}\right).$$

Since the curvature of γ is everywhere less than or equal to $\kappa_{\max}$, the curve cannot bend more sharply than the osculating circle with curvature $\kappa_{\max}$. Therefore, the maximum deviation $d_{\max}$ between γ and its chord cannot exceed that of the osculating circle:

$$\operatorname{dist}(\boldsymbol{w}, L^{-1}(c)) \leq d_{\max} \stackrel{\text{def}}{=} \frac{1}{\kappa_{\max}}\left(1 - \sqrt{1 - \left(\frac{\kappa_{\max}\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2}{2}\right)^2}\right).$$

Assuming L is Lipschitz continuous with Lipschitz constant C_L, for any $\boldsymbol{w}$ on S, we have

$$|L(\boldsymbol{w}) - c| = |L(\boldsymbol{w}) - L(\gamma(t))| \leq C_L\|\boldsymbol{w} - \gamma(t)\| \leq C_L d_{\max}.$$

□

Appendix C

Supplementary Material for Chapter 4

C.1 Group Actions

In this section, we derive the group actions for the test functions and multi-layer neural networks. More details about group theory can be found in textbooks such as [87].

C.1.1 Continuous Symmetry in Test Functions

Ellipse

Consider the following loss function with $a \in \mathbb{R}^{\geq 0}$:

$$L(x_1, x_2) = x_1^2 + a x_2^2 \tag{C.1}$$

If we change the variables to $L(u(x_1,x_2), v(x_1,x_2)) = u^2 + v^2$, 2D rotations leave L unchanged. Therefore $SO(2)$ is a symmetry of $L(x_1,x_2)$. Let $g_\theta \in SO(2)$, and define the group action as

$$g_\theta \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/\sqrt{a} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{a} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{C.2}$$

Then

$$L(x_1, x_2) = L(g \cdot (x_1, x_2)) \tag{C.3}$$

Rosenbrock function

Consider the Rosenbrock function with 2 variables [120]:

$$L(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \tag{C.4}$$

Let $u = 10(x_1^2 - x_2)$ and $v = x_1 - 1$. After changing the variables from x and y to u and v, L has a rotational symmetry. Note that the function, $h : \mathbb{R}^2 \to \mathbb{R}^2$, that maps x_1, x_2 to u, v is bijective:

$$\begin{aligned} (u, v) &= h(x_1, x_2) = (10(x_1^2 - x_2), x_1 - 1) \\ (x_1, x_2) &= h^{-1}(u, v) = (v + 1, (v + 1)^2 - 0.1u) \\ h(x_1, x_2) &= h(y_1, y_2) \Rightarrow (x_1, x_2) = (y_1, y_2) \end{aligned} \tag{C.5}$$

Next, we show that $SO(2)$ is a symmetry of $L(x_1, x_2)$. Let ρ be a representation of $SO(2)$ acting on $\mathbb{R}^2$. For $g \in SO(2)$, define the following group action:

$$g \cdot (x_1, x_2) = h^{-1}(\rho(g) h(x_1, x_2)) \tag{C.6}$$

Then

$$L(x_1, x_2) = L(g \cdot (x_1, x_2)) \tag{C.7}$$

For the Rosenbrock function with 2N parameters, we can construct a bijective function $h : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ by transforming each of the N pairs of variables as before, and $SO(2N)$ is a

symmetry of $L(x_1, ..., x_{2N})$. However, we will only use the 2 variable version in the experiments.

Booth function

Consider the Booth function [71]:

$$L(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Similar to the Rosebrock function, a change of variables reveals a rotational symmetry of L:

$$\begin{aligned}(u, v) &= h(x_1, x_2) = (x_1 + 2x_2 - 7, 2x_1 + x_2 - 5) \\ (x_1, x_2) &= h^{-1}(u, v) = (-\frac{1}{3}u + \frac{2}{3}v + 1, \frac{2}{3}u - \frac{1}{3}v + 3)\end{aligned} \tag{C.8}$$

The function $h : \mathbb{R}^2 \to \mathbb{R}^2$ that maps x_1, x_2 to u, v is bijective. Let ρ be a representation of $SO(2)$ acting on $\mathbb{R}^2$. For $g \in SO(2)$, define the following group action:

$$g \cdot (x_1, x_2) = h^{-1}(\rho(g) h(x_1, x_2)) \tag{C.9}$$

Then $L(x_1, x_2)$ admits an $SO(2)$ symmetry:

$$L(x_1, x_2) = L(g \cdot (x_1, x_2)) \tag{C.10}$$

C.1.2 Continuous Symmetry in Multi-layer Neural Networks

In this and the following sections, we provide proofs for the theoretical results. We restate the propositions from the main text for readability.

Proposition C.1.1. *A linear network is invariant under all groups $G_m \equiv \mathrm{GL}_{d_m}$ acting as*

$$g \cdot (W_m, W_{m-1}) = (W_m g^{-1}, g W_{m-1}), \qquad g \cdot W_k = W_k, \quad \forall k \notin \{m, m-1\}.$$

Proof. In the linear network $h_m = W_m h_{m-1}$, hence

$$g \cdot (W_m, h_{m-1}) = (W_m g^{-1}, g h_{m-1}), \quad g \cdot h_m = W_m g^{-1} g h_{m-1} = h_m \tag{C.11}$$

which means a p-layer linear network is invariant under all G_m with $m \leq p$ as they keep the output h_p invariant ($\forall g \in G_m$, $g \cdot h_p = h_p$). □

Proposition 4.4.2. *Assume that h_{m-2} is invertible. A multi-layer network with bijective activation σ has a $\mathrm{GL}_{d_{m-1}}$ symmetry. For $g_m \in G_m = \mathrm{GL}_{d_{m-1}}(\mathbb{R})$ the following group action keeps h_p with $p \geq m$ invariant*

$$g_m \cdot W_k = \begin{cases} W_m g_m^{-1} & k = m \\ \sigma^{-1}\left(g_m \sigma\left(W_{m-1} h_{m-2}\right)\right) h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases} \tag{C.12}$$

Proof. We want to convert $g \cdot h_{m-1}$ into a transformation on W_{m-1} instead of h_{m-1}. In other words, we want to find a set of transformed weights W'_m, W'_{m-1} which yields the same network output $\tilde{h}_m$:

$$\begin{aligned} \tilde{h}_m = W'_m \sigma\left(W'_{m-1} h_{m-2}\right) &= W_m g^{-1} g \sigma\left(W_{m-1} h_{m-2}\right) \\ \Rightarrow W'_m = W_m g^{-1}, \qquad \sigma\left(W'_{m-1} h_{m-2}\right) &= g \sigma\left(W_{m-1} h_{m-2}\right) \end{aligned} \tag{C.13}$$

Solving equation C.13 we get

$$W'_{m-1} = \sigma^{-1}\left(g \sigma\left(W_{m-1} h_{m-2}\right)\right) h_{m-2}^{-1}. \tag{C.14}$$

equation C.12 follows from equation C.13 and equation C.14.

To verify that equation C.12 is a valid group action,

$$I \cdot W_k = \begin{cases} W_m I & k = m \\ \sigma^{-1}\left(I\sigma\left(W_{m-1}h_{m-2}\right)\right)h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases}$$
$$= W_k \tag{C.15}$$

and

$$g_1 \cdot (g_2 \cdot W_k) = \begin{cases} W_m g_2^{-1} g_1^{-1} & k = m \\ \sigma^{-1}\left(g_1\sigma\left(\left[\sigma^{-1}\left(g_2\sigma\left(W_{m-1}h_{m-2}\right)\right)h_{m-2}^{-1}\right]h_{m-2}\right)\right)h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases}$$
$$= \begin{cases} W_m(g_1 g_2)^{-1} & k = m \\ \sigma^{-1}\left((g_1 g_2)\sigma\left(W_{m-1}h_{m-2}\right)\right)h_{m-2}^{-1} & k = m-1 \\ W_k & k \notin \{m, m-1\} \end{cases}$$
$$= (g_1 g_2) \cdot W_k$$

(C.16)

□

C.2 Teleportation and SGD

This section includes a proof for Theorem 4.5.9. Additionally, we discuss the theorem's implication when the loss function is strictly convex.

Lemma C.2.1 (Descent Lemma)**.** *Let $L(\boldsymbol{w}, \xi)$ be a β–smooth function. It follows that*

$$\mathbb{E}\left[\|\nabla L(\boldsymbol{w}, \xi)\|^2\right] \leq 2\beta(L(\boldsymbol{w}) - L(\boldsymbol{w}^*)) + 2\beta(L(\boldsymbol{w}^*) - \mathbb{E}\left[\inf_{\boldsymbol{w}} L(\boldsymbol{w}, \xi)\right]). \tag{C.17}$$

Proof. Since $L(w,\xi)$ is smooth we have that

$$L(z,\xi) - L(\boldsymbol{w},\xi) \leq \langle \nabla L(\boldsymbol{w},\xi), \boldsymbol{z}-\boldsymbol{w}\rangle + \frac{\beta}{2}\|\boldsymbol{z}-\boldsymbol{w}\|^2, \quad \forall \boldsymbol{z},\boldsymbol{w} \in \mathbb{R}^d. \tag{C.18}$$

By inserting

$$\boldsymbol{z} = \boldsymbol{w} - \frac{1}{\beta}\nabla L(\boldsymbol{w},\xi)$$

into equation C.18 we have that

$$L\big(\boldsymbol{w} - (1/\beta)\nabla L(\boldsymbol{w},\xi),\xi\big) \leq L(\boldsymbol{w},\xi) - \frac{1}{2\beta}\|\nabla L(\boldsymbol{w},\xi)\|^2. \tag{C.19}$$

Re-arranging we have that

$$\begin{aligned}
L(\boldsymbol{w}^*,\xi) - L(\boldsymbol{w},\xi) &= L(\boldsymbol{w}^*,\xi) - \inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) + \inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) - L(\boldsymbol{w},\xi) \\
&\leq L(\boldsymbol{w}^*,\xi) - \inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) + L\big(\boldsymbol{w} - (1/\beta)\nabla L(\boldsymbol{w},\xi),\xi\big) - L(\boldsymbol{w},\xi) \\
&\overset{equation\ C.19}{\leq} L(\boldsymbol{w}^*,\xi) - \inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) - \frac{1}{2\beta}\|\nabla L(\boldsymbol{w},\xi)\|^2,
\end{aligned}$$

where the first inequality follows because $\inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) \leq L(\boldsymbol{w},\xi), \forall \boldsymbol{w}$. Re-arranging the above and taking expectation gives

$$\begin{aligned}
\mathbb{E}\left[\|\nabla L(\boldsymbol{w},\xi)\|^2\right] &\leq 2\mathbb{E}\left[\beta\left(L(\boldsymbol{w}^*,\xi) - \inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) + L(\boldsymbol{w},\xi) - L(\boldsymbol{w}^*,\xi)\right)\right] \\
&\leq 2\beta\mathbb{E}\left[L(\boldsymbol{w}^*,\xi) - \inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi) + L(\boldsymbol{w},\xi) - L(\boldsymbol{w}^*,\xi)\right] \\
&\leq 2\beta(L(\boldsymbol{w}) - L(\boldsymbol{w}^*)) + 2\beta(L(\boldsymbol{w}^*) - \mathbb{E}\left[\inf_{\boldsymbol{w}} L(\boldsymbol{w},\xi)\right]).
\end{aligned}$$

□

At each iteration $t \in \mathbb{N}^+$ in SGD, we choose a group element $g^t \in G$ and use teleportation

before each gradient step as follows

$$\boldsymbol{w}^{t+1} = g^t \cdot \boldsymbol{w}^t - \eta \nabla L(g^t \cdot \boldsymbol{w}^t, \xi^t). \tag{C.20}$$

Here η is a learning rate, $\nabla L(\boldsymbol{w}^t, \xi^t)$ is a gradient of $L(\boldsymbol{w}^t, \xi^t)$ with respect to the parameters $\boldsymbol{w}$, and $\xi^t \sim \mathscr{D}$ is a mini-batch of data sampled i.i.d at each iteration.

Theorem 4.5.9. *Let $L(\boldsymbol{w}, \xi)$ be β–smooth and let*

$$\sigma^2 \stackrel{def}{=} L(\boldsymbol{w}^*) - \mathbb{E}\left[\inf_{\boldsymbol{w}} L(\boldsymbol{w}, \xi)\right].$$

Consider the iterates $\boldsymbol{w}^t$ given by equation 4.45 where

$$g^t \in \arg\max_{g \in G} \|\nabla L(g \cdot \boldsymbol{w}^t)\|^2. \tag{C.21}$$

If $\eta = \frac{1}{\beta\sqrt{T-1}}$ then

$$\min_{t=0,\ldots,T-1} \mathbb{E}\left[\max_{g \in G} \|\nabla L(g \cdot \boldsymbol{w}^t)\|^2\right] \leq \frac{2\beta}{\sqrt{T-1}} \mathbb{E}\left[L(\boldsymbol{w}^0) - L(\boldsymbol{w}^*)\right] + \frac{\beta\sigma^2}{\sqrt{T-1}}. \tag{C.22}$$

Proof. First note that if $L(\boldsymbol{w}, \xi)$ is β–smooth, then $\mathscr{L}(\boldsymbol{w})$ is also a β–smooth function, that is

$$L(\boldsymbol{z}) - L(\boldsymbol{w}) - \langle \nabla L(\boldsymbol{w}), \boldsymbol{z} - \boldsymbol{w} \rangle \leq \frac{\beta}{2} \|\boldsymbol{z} - \boldsymbol{w}\|^2. \tag{C.23}$$

Using equation 4.45 with $\boldsymbol{z} = \boldsymbol{w}^{t+1}$ and $\boldsymbol{w} = g^t \cdot \boldsymbol{w}^t$, together with equation C.23 and the fact that the group action preserves loss, we have that

$$L(\boldsymbol{w}^{t+1}) \leq L(g^t \cdot \boldsymbol{w}^t) + \left\langle \nabla L(g^t \cdot \boldsymbol{w}^t), \boldsymbol{w}^{t+1} - g^t \cdot \boldsymbol{w}^t \right\rangle + \frac{\beta}{2} \|\boldsymbol{w}^{t+1} - g^t \cdot \boldsymbol{w}^t\|^2 \tag{C.24}$$

$$= L(\boldsymbol{w}^t) - \eta_t \left\langle \nabla L(g^t \cdot \boldsymbol{w}^t), \nabla L(g^t \cdot \boldsymbol{w}^t, \xi^t) \right\rangle + \frac{\beta \eta_t^2}{2} \|\nabla L(g^t \cdot \boldsymbol{w}^t, \xi^t)\|^2. \tag{C.25}$$

Taking expectation conditioned on $\boldsymbol{w}^t$, we have that

$$\mathbb{E}_t\left[L(\boldsymbol{w}^{t+1})\right] \leq L(\boldsymbol{w}^t) - \eta_t \|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2 + \frac{\beta \eta_t^2}{2} \mathbb{E}_t\left[\|\nabla L(g^t \cdot \boldsymbol{w}^t, \xi^t)\|^2\right]. \tag{C.26}$$

Now since $L(\boldsymbol{w}, \xi)$ is β–smooth, from Lemma C.2.1 above we have that

$$\mathbb{E}\left[\|\nabla L(\boldsymbol{w}, \xi)\|^2\right] \leq 2\beta(L(\boldsymbol{w}) - L(\boldsymbol{w}^*)) + 2\beta(L(\boldsymbol{w}^*) - \mathbb{E}\left[\inf_{\boldsymbol{w}} L(\boldsymbol{w}, \xi)\right]) \tag{C.27}$$

Using equation C.27 with $\boldsymbol{w} = g^t \circ \boldsymbol{w}^t$ we have that

$$\begin{aligned}\mathbb{E}_t\left[L(\boldsymbol{w}^{t+1})\right] \leq L(\boldsymbol{w}^t) &- \eta_t \|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2 \\ &+ \beta^2 \eta_t^2 \left(L(g^t \cdot \boldsymbol{w}^t) - L(\boldsymbol{w}^*) + L(\boldsymbol{w}^*) - \mathbb{E}\left[\inf_{\boldsymbol{w}} L(\boldsymbol{w}, \xi)\right]\right). \end{aligned} \tag{C.28}$$

Using that $L(g^t \cdot \boldsymbol{w}^t) = L(\boldsymbol{w}^t)$, taking full expectation and re-arranging terms gives

$$\eta_t \mathbb{E}\left[\|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2\right] \leq (1 + \beta^2 \eta_t^2)\mathbb{E}\left[L(\boldsymbol{w}^t) - L^*\right] - \mathbb{E}\left[L(\boldsymbol{w}^{t+1}) - L^*\right] + \beta^2 \eta_t^2 \sigma^2. \tag{C.29}$$

Now we use a re-weighting trick introduced in [133]. Let $\alpha_t > 0$ be a sequence such that $\alpha_t(1 + \beta^2 \eta_t^2) = \alpha_{t-1}$. Consequently if $\alpha_{-1} = 1$ then $\alpha_t = (1 + \beta^2 \eta_t^2)^{-(t+1)}$. Multiplying by both sides of equation C.29 by α_t thus gives

$$\alpha_t \eta_t \mathbb{E}\left[\|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2\right] \leq \alpha_{t-1}\mathbb{E}\left[L(\boldsymbol{w}^t) - L^*\right] - \alpha_t \mathbb{E}\left[L(\boldsymbol{w}^{t+1}) - L^*\right] + \alpha_t \beta^2 \eta_t^2 \sigma^2. \tag{C.30}$$

Summing up from $t = 0, \ldots, T-1$, and using telescopic cancellation, gives

$$\sum_{t=0}^{T-1} \alpha_t \eta_t \mathbb{E}\left[\|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2\right] \leq \mathbb{E}\left[L(\boldsymbol{w}^0) - L^*\right] + \beta^2 \sigma^2 \sum_{t=0}^{T-1} \alpha_t \eta_t^2 \tag{C.31}$$

Let $A = \sum_{t=0}^{T-1} \alpha_t \eta_t$. Dividing both sides by A gives

$$\min_{t=0,\dots,T-1} \mathbb{E}\left[\|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2\right] \leq \frac{1}{\sum_{t=0}^{T-1} \alpha_t \eta_t} \sum_{t=0}^{T-1} \alpha_t \eta_t \|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2$$
$$\leq \frac{\mathbb{E}\left[L(\boldsymbol{w}^0) - L^*\right] + \beta^2 \sigma^2 \sum_{t=0}^{T-1} \alpha_t \eta_t^2}{\sum_{t=0}^{T-1} \alpha_t \eta_t}. \tag{C.32}$$

Finally, if $\eta_t \equiv \eta$ then

$$\sum_{t=0}^{T-1} \alpha_t \eta_t = \eta \sum_{t=0}^{T-1} (1+\beta^2 \eta_t^2)^{-(t+1)} = \frac{\eta}{1+\beta^2\eta^2} \frac{1-(1+\beta^2\eta^2)^{-T}}{1-(1+\beta^2\eta^2)^{-1}} \tag{C.33}$$
$$= \frac{1-(1+\beta^2\eta^2)^{-T}}{\beta^2 \eta} \tag{C.34}$$

To bound the term with the $-T$ power, we use that

$$(1+\beta^2\eta^2)^{-T} \leq \frac{1}{2} \quad \Longrightarrow \quad \frac{\log(2)}{\log(1+\beta^2\eta^2)} \leq T.$$

To simplify the above expression we can use

$$\frac{x}{1+x} \leq \log(1+x) \leq x, \quad \text{for } x \geq -1,$$

thus

$$\frac{\log(2)}{\log(1+\beta^2\eta^2)} \leq \frac{1+\beta^2\eta^2}{\beta^2\eta^2} \leq T.$$

Using the above we have that

$$\sum_{t=0}^{T-1} \alpha_t \eta_t \geq \frac{1}{2\beta^2\eta}, \quad \text{for } T \geq \frac{1+\beta^2\eta^2}{\beta^2\eta^2}$$

Using this lower bound in equation C.32 gives

$$\min_{t=0,\dots,T-1} \mathbb{E}\left[\|\nabla L(g^t \cdot \boldsymbol{w}^t)\|^2\right] \leq 2\beta^2\eta\, \mathbb{E}\left[L(\boldsymbol{w}^0) - L^*\right] + \eta\beta^2\sigma^2, \quad \text{for } T \geq \frac{1+\beta^2\eta^2}{\beta^2\eta^2}.$$

Now note that

$$T \geq \frac{1+\beta^2\eta^2}{\beta^2\eta^2} \Leftrightarrow \beta^2\eta^2(T-1) \geq 1 \Leftrightarrow \eta \geq \frac{1}{\beta\sqrt{(T-1)}}.$$

Thus finally setting $\eta = \frac{1}{\beta\sqrt{T-1}}$ gives the result equation 4.46.

□

Proposition C.2.2. *Assume that $L : \mathbb{R}^n \to \mathbb{R}$ is strictly convex and twice continuously differentiable. Assume also that for any two points $\boldsymbol{w}_a, \boldsymbol{w}_b \in \mathbb{R}^n$ such that $L(\boldsymbol{w}_a) = L(\boldsymbol{w}_b)$, there exists a $g \in G$ such that $\boldsymbol{w}_a = g \cdot \boldsymbol{w}_b$. At two points $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^n$, if $\max_{g\in G}\|\nabla L(g \cdot \boldsymbol{w}_1)\|^2 = \|\nabla L(\boldsymbol{w}_2)\|^2$, then $L(\boldsymbol{w}_1) \leq L(\boldsymbol{w}_2)$.*

Proof. Let $S(x) = \{\boldsymbol{w} : L(\boldsymbol{w}) = x\}$ be the level sets of L, and $X = \{L(\boldsymbol{w}) : \boldsymbol{w} \in \mathbb{R}^n\}$ be the image of L. Since G acts transitively on the level sets of L, $\max_{g\in G}\|\nabla L(g \cdot \boldsymbol{w})\|^2 = \max_{\boldsymbol{w}\in S(x)}\|\nabla L(\boldsymbol{w})\|^2$. To simplify notation, we define a function $F : X \to \mathbb{R}$, $F(x) = \max_{\boldsymbol{w}\in S(x)}\|\nabla L(\boldsymbol{w})\|^2$. Since $\nabla L(\boldsymbol{w})$ is continuously differentiable, the directional derivative of F is defined. Additionally, since L is continuous and its domain $\mathbb{R}^n$ is connected, its image X is also connected. This means that for any $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^n$ and $\min(L(\boldsymbol{w}_1), L(\boldsymbol{w}_2)) \leq y \leq \max(L(\boldsymbol{w}_1), L(\boldsymbol{w}_2))$, there exists a $\boldsymbol{w}_3 \in \mathbb{R}^n$ such that $L(\boldsymbol{w}_3) = y$.

Next, we show that $F(\cdot)$ is strictly increasing by contradiction.

Suppose that $L(\boldsymbol{w}_1) < L(\boldsymbol{w}_2)$ and $F(L(\boldsymbol{w}_1)) \geq F(L(\boldsymbol{w}_2))$. By the mean value theorem, there exists a $\boldsymbol{w}_3$ such that $L(\boldsymbol{w}_1) < L(\boldsymbol{w}_3) < L(\boldsymbol{w}_2)$ and the directional derivative of F in the direction towards $L(\boldsymbol{w}_2)$ is non-positive, that is, $\partial_{L(\boldsymbol{w}_2)-L(\boldsymbol{w}_3)}F(L(\boldsymbol{w}_3)) \leq 0$. Let $\boldsymbol{w}_3^* \in \arg\max_{\boldsymbol{w}\in S(L(\boldsymbol{w}_3))}\|\nabla L(\boldsymbol{w})\|^2$ be a point that has the largest gradient norm in $S(L(\boldsymbol{w}_3))$. Then at $\boldsymbol{w}_3^*$, $\|\nabla L\|^2$ cannot increase along the gradient direction. However, this means

$$\nabla L(\boldsymbol{w}_3^*) \cdot \frac{\partial}{\partial \boldsymbol{w}}\|\nabla L(\boldsymbol{w}_3^*)\|^2 = \nabla L(\boldsymbol{w}_3^*)^T H \nabla L(\boldsymbol{w}_3^*) \leq 0. \tag{C.35}$$

Since we assumed that L is convex and $L(\boldsymbol{w}_3^*)$ is not a minimum ($L(\boldsymbol{w}_3^*) > L(\boldsymbol{w}_1)$), we have that

$\nabla L(\boldsymbol{w}_3^*) \neq 0$. Therefore, equation C.35 contradicts with L being strictly convex, and we have $F(L(\boldsymbol{w}_1)) < F(L(\boldsymbol{w}_2))$.

We have shown that $L(\boldsymbol{w}_1) < L(\boldsymbol{w}_2)$ implies $F(L(\boldsymbol{w}_1)) < F(L(\boldsymbol{w}_2))$. Taking the contrapositive and switching $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, $F(L(\boldsymbol{w}_1)) \leq F(L(\boldsymbol{w}_2))$ implies $L(\boldsymbol{w}_1) \leq L(\boldsymbol{w}_2)$. Equivalently, $\max_{g\in G}\|\nabla L(g\cdot \boldsymbol{w}_1)\|^2 \leq \max_{g\in G}\|\nabla L(g\cdot \boldsymbol{w}_2)\|^2$ implies that $L(\boldsymbol{w}_1) \leq L(\boldsymbol{w}_2)$.

Finally, since

$$\max_{g\in G}\|\nabla L(g\cdot \boldsymbol{w}_1)\|^2 = \|\nabla L(\boldsymbol{w}_2)\|^2 \leq \max_{g\in G}\|\nabla L(g\cdot \boldsymbol{w}_2)\|^2, \tag{C.36}$$

we have $L(\boldsymbol{w}_1) \leq L(\boldsymbol{w}_2)$. □

C.3 Teleportation and Newton's Method

Lemma C.3.1 (One step of Newton's Method)**.** *Let $f(x)$ be a μ–strongly convex and L–smooth function, that is, we have a global lower bound on the Hessian given by*

$$LI \succeq \nabla^2 f(x) \succeq \mu I, \quad \forall x \in \mathbb{R}^n. \tag{C.37}$$

Furthermore, if the Hessian is also G–Lipschitz

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq G\|x-y\| \tag{C.38}$$

then Newton's method

$$x^{k+1} = x^k - \lambda_k \nabla^2 f(x^k)^{-1} \nabla f(x^k)$$

has a mixed linear and quadratic convergence according to

$$\|x^{k+1} - x^*\| \leq \frac{G}{2\mu}\|x^k - x^*\|^2 + |1-\lambda_k|\frac{L}{2\mu}\|x^k - x^*\|. \tag{C.39}$$

Proof. By mean value theorem,

$$\begin{aligned}
x^{k+1} - x^* &= x^k - x^* - \lambda_k \nabla^2 f(x^k)^{-1} \left(\nabla f(x^k) - \nabla f(x^*) \right) \\
&= x^k - x^* - \lambda_k \nabla^2 f(x^k)^{-1} \int_{s=0}^{1} \nabla^2 f(x^k + s(x^* - x^k))(x^k - x^*) ds \\
&= \nabla^2 f(x^k)^{-1} \int_{s=0}^{1} \left(\nabla^2 f(x^k) - \lambda_k \nabla^2 f(x^k + s(x^* - x^k)) \right) (x^k - x^*) ds \\
&= \nabla^2 f(x^k)^{-1} \int_{s=0}^{1} \Big(\nabla^2 f(x^k) - \nabla^2 f(x^k + s(x^* - x^k)) \\
&\qquad + (1 - \lambda_k) \nabla^2 f(x^k + s(x^* - x^k)) \Big) (x^k - x^*) ds
\end{aligned}$$

Let $\delta_k := \|x^{k+1} - x^*\|$. Taking norms we have that

$$\begin{aligned}
\delta_{k+1} &\leq \|\nabla^2 f(x^k)^{-1}\| \int_{s=0}^{1} \Big(\|\nabla^2 f(x^k) - \nabla^2 f(x^k + s(x^* - x^k))\| \\
&\qquad + |1 - \lambda_k| \|\nabla^2 f(x^k + s(x^* - x^k))\| \Big) \delta_k ds \\
&\overset{\text{equation C.38+equation C.37}}{\leq} \frac{G}{\mu} \int_{s=0}^{1} s \|x^k - x^*\|^2 ds + |1 - \lambda_k| \frac{L}{\mu} \int_{s=0}^{1} s \|x^k - x^*\| ds \\
&= \frac{G}{2\mu} \|x^k - x^*\|^2 + |1 - \lambda_k| \frac{L}{2\mu} \|x^k - x^*\|.
\end{aligned}$$

□

The assumptions on for this proof can be relaxed, since we only require the Hessian is Lipschitz and lower bounded in a $\frac{\mu}{2L}$–ball around x^*.

Proposition 4.5.12 (Quadratic term in convergence rate)**.** *Let L be strictly convex and let $w_0 \in \mathbb{R}^d$. Let*

$$w' \in \arg\max_{w \in \mathbb{R}^d} \frac{1}{2} \|\nabla L(w)\|^2 \quad \textit{subject to} \quad L(w) = L(w_0). \tag{C.40}$$

If $\nabla L(w') \neq 0$ then there exists λ_0 such that

$$0 \leq \lambda_0 \leq \lambda_{\max}(\nabla^2 L(w_0))$$

and one step of gradient descent with learning rate $\gamma > 0$ gives

$$\begin{aligned} w_1 &= w' - \gamma \nabla L(w') \\ &= w' - \gamma \lambda_0 \nabla^2 L(w')^{-1} \nabla L(w'). \end{aligned} \tag{C.41}$$

Consequently, letting $w' = g_0 \circ w_0$, and if $\gamma \leq \frac{1}{\lambda_0}$ then under the assumptions of Lemma C.3.1 we have that

$$\|w_1 - w^*\| \leq \frac{G}{2\mu} \|g_0 \circ w_0 - x^*\|^2 + |1 - \gamma\lambda_0| \frac{L}{2\mu} \|g_0 \circ w_0 - w^*\|.$$

Proof. The Lagrangian associated to equation C.40 is given by

$$L(w, \lambda) = \frac{1}{2} \|\nabla L(w)\|^2 + \lambda (L(w_0) - L(w)).$$

Taking the derivative in w and setting it to zero gives

$$\nabla_w L(w, \lambda_0) = 0 \implies \nabla^2 L(w) \nabla L(w) - \lambda_0 \nabla L(w) = 0. \tag{C.42}$$

Re-arranging we have that

$$\nabla L(w) = \lambda_0 \nabla^2 L(w)^{-1} \nabla L(w).$$

If $\nabla L(w') \neq 0$ then from the above we have that

$$\|\nabla L(w)\|^2 = \lambda_0 \nabla L(w)^\top \nabla^2 L(w)^{-1} \nabla L(w) > 0.$$

Since $\nabla^2 L(w)^{-1}$ is positive definite we have that $\nabla L(w)^\top \nabla^2 L(w)^{-1} \nabla L(w) \geq 0$, and consequently $\lambda_0 > 0$. Finally from equation C.42 we have that λ_0 is an eigenvalue of $\nabla^2 L(w)$ and thus it must be smaller or equal to the largest eigenvalue of $\nabla^2 L(w)$.

□

C.4 Is One Teleportation Enough to Find the Optimal Trajectory?

Lemma C.4.1. *For two vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$, if $\boldsymbol{v}^T \boldsymbol{w} = 0$ and $\boldsymbol{w} \neq \mathbf{0}$, then there exists an anti-symmetric matrix $M \in \mathbb{R}^{n \times n}$ such that $\boldsymbol{v} = M\boldsymbol{w}$.*

Proof. Let $\boldsymbol{w}_0 = [1, 0, ..., 0]^T \in \mathbb{R}^n$. Consider a list of $n-1$ anti-symmetric matrices $M_i \in \mathbb{R}^{n \times n}$, where

$$M_{ij}^k = \begin{cases} -1, & \text{if } j = 1 \text{ and } k = i+1 \\ 1, & \text{if } j = i+1 \text{ and } k = 1 \\ 0, & \text{otherwise} \end{cases} \tag{C.43}$$

In matrix form, the M_i's are

$$M_1 = \begin{bmatrix} 0 & -1 & 0 & ... & 0 \\ 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & ... & 0 \\ & & ... & & \\ 0 & 0 & 0 & ... & 0 \end{bmatrix}, M_2 = \begin{bmatrix} 0 & 0 & -1 & ... & 0 \\ 0 & 0 & 0 & ... & 0 \\ 1 & 0 & 0 & ... & 0 \\ & & ... & & \\ 0 & 0 & 0 & ... & 0 \end{bmatrix}, ..., M_{n-1} = \begin{bmatrix} 0 & 0 & 0 & ... & -1 \\ 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & ... & 0 \\ & & ... & & \\ 1 & 0 & 0 & ... & 0 \end{bmatrix}. \tag{C.44}$$

Since M_i's are anti-symmetric, $M_i \boldsymbol{w}_0$ is orthogonal to $\boldsymbol{w}_0$. The norm of $M_i \boldsymbol{w}_0 = \mathbf{e}_{i+1}$ is 1. Additionally, $M_i \boldsymbol{w}_0$ is orthogonal to $M_j \boldsymbol{w}_0$ for $i \neq j$:

$$(M_i \boldsymbol{w}_0)^T (M_j \boldsymbol{w}_0) = \mathbf{e}_{i+1}^T \mathbf{e}_{j+1} = \delta_{ij}. \tag{C.45}$$

Denote $\boldsymbol{w}_0^\perp = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x}^T \boldsymbol{w}_0 = 0\}$ as the orthogonal complement of $\boldsymbol{w}_0$. Then $M_i \boldsymbol{w}_0$ forms a basis of $\boldsymbol{w}_0^\perp$. Next, we extend this to an arbitrary $\boldsymbol{w} \in \mathbb{R}^n$.

Let $\hat{\boldsymbol{w}} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$. Since $\hat{\boldsymbol{w}}$ has norm 1, there exists an orthogonal matrix R such that $\hat{\boldsymbol{w}} = R\boldsymbol{w}_0$. Let $M_i' = RM_iR^T$. Then M_i' is anti-symmetric:

$$(RM_iR^T)^T = RM_i^TR^T = -RM_iR^T. \tag{C.46}$$

It follows that $M_i'\hat{\boldsymbol{w}}$ is orthogonal to $\hat{\boldsymbol{w}}$. The norm of $M_i'\hat{\boldsymbol{w}}$ is $\|(RM_iR^T)(R\boldsymbol{w}_0)\| = \|RM_i\boldsymbol{w}_0\| = \|M_i\boldsymbol{w}_0\| = 1$. Additionally, $M_i'\hat{\boldsymbol{w}}$ is orthogonal to $M_j'\hat{\boldsymbol{w}}$ for $i \neq j$:

$$\begin{aligned}(M_i'\hat{\boldsymbol{w}})^T(M_j'\hat{\boldsymbol{w}}) &= (RM_iR^TR\boldsymbol{w}_0)^T(RM_jR^TR\boldsymbol{w}_0)\\ &= \boldsymbol{w}_0^TR^TRM_i^TR^TRM_jR^TR\boldsymbol{w}_0\\ &= \boldsymbol{w}_0^TM_i^TM_j\boldsymbol{w}_0\\ &= \delta_{ij}.\end{aligned} \tag{C.47}$$

Therefore, $M_i'\hat{\boldsymbol{w}}$ spans $\hat{\boldsymbol{w}}^\perp = \boldsymbol{w}^\perp$. This means that any vector $\boldsymbol{v} \in \boldsymbol{w}^\perp$ can be written as a linear combination of $M_i'\hat{\boldsymbol{w}}$. That is, there exists $k_1, ..., k_n \in \mathbb{R}$, such that $\boldsymbol{v} = \sum_i k_i(M_i'\hat{\boldsymbol{w}})$. To find the anti-symmetric M that takes $\boldsymbol{w}$ to $\boldsymbol{v}$, note that

$$\boldsymbol{v} = \left(\sum_i k_iM_i'\right)\hat{\boldsymbol{w}} = \left(\|\boldsymbol{w}\|_2^{-1}\sum_i k_iM_i'\right)\boldsymbol{w}. \tag{C.48}$$

Since the sum of anti-symmetric matrices is anti-symmetric, and the product of an anti-symmetric matrix and a scalar is also anti-symmetric, $\|\boldsymbol{w}\|_2^{-1}\sum_i k_iM_i'$ is anti-symmetric. □

Lemma C.4.2. *Let $\boldsymbol{v} \in \mathbb{R}^n$ be a nonzero vector. Then the two sets $\{M\boldsymbol{v} : M \in \mathbb{R}^{n\times n}, M^T = -M\}$ and $\{\boldsymbol{w} \in \mathbb{R}^n : \boldsymbol{w}^T\boldsymbol{v} = 0\}$ are equal.*

Proof. Let $A = \{M\boldsymbol{v} : M \in \mathbb{R}^{n\times n}, M^T = M^{-1}\}$ and $B = \{\boldsymbol{w} \in \mathbb{R}^n : \boldsymbol{w}^T\boldsymbol{v} = 0\}$. Since $(M\boldsymbol{v})^T\boldsymbol{v} = 0$ for all anti-symmetric M, every element in A is in B. By Lemma C.4.1, every element in B is in A. Therefore $A = B$. □

Let $S = \{(M\frac{\partial L}{\partial \boldsymbol{w}})^i\frac{\partial}{\partial w^i} \in \mathfrak{X} |\ M \in \mathbb{R}^{n\times n}, M^T = -M\}$ be the set of vector fields constructed

by multiplying the gradient by an anti-symmetric matrix. Recall that $R = -\frac{\partial L}{\partial w_i}\frac{\partial}{\partial w^i}$ is the reverse gradient vector field, and $\mathfrak{X}_\perp = \{a^i \frac{\partial}{\partial w^i} |\ \sum_i a^i(\boldsymbol{w})\frac{\partial L(\boldsymbol{w})}{\partial w^i} = 0, \forall \boldsymbol{w} \in \mathscr{M}\}$ is the set of all vector fields orthogonal to R. From Lemma C.4.2, we have $S = \mathfrak{X}_\perp$. Therefore, a point $\boldsymbol{w}$ is an optimal point in S if and only if $\boldsymbol{w}$ is an optimal point in $\mathfrak{X}_\perp$.

We are now ready to prove the following proposition, which provides another way to check the condition in Proposition 4.5.15.

Proposition 4.5.16. *If at all optimal points in S,*

$$M^j_\alpha \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} = 0 \tag{C.49}$$

for all anti-symmetric matrix $M \in \mathbb{R}^{n\times n}$, then the gradient flow starting at an optimal point in S is optimal in S.

Proof. Expanding $R[A,R]L$, we have

$$\begin{aligned}
R[A,R]L &= R\left(A\left(r^i\frac{\partial L}{\partial w^i}\right) - 0\right)\\
&= r^k\frac{\partial}{\partial w^k}\left(a^j\frac{\partial}{\partial w_j}\left(r^i\frac{\partial L}{\partial w^i}\right)\right)\\
&= r^k\frac{\partial}{\partial w^k}\left(a^j\left(\frac{\partial r^i}{\partial w^j}\frac{\partial L}{\partial w^i} + r^i\frac{\partial}{\partial w^j}\frac{\partial L}{\partial w^i}\right)\right)\\
&= -r^k\frac{\partial}{\partial w^k}\left(a^j\left(\left(\frac{\partial}{\partial w^j}\frac{\partial L}{\partial w_i}\right)\frac{\partial L}{\partial w^i} + \frac{\partial L}{\partial w_i}\frac{\partial}{\partial w^j}\frac{\partial L}{\partial w^i}\right)\right)\\
&= -2r^k\frac{\partial}{\partial w^k}\left(a^j\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i}\right)\\
&= -2r^k\left(\frac{\partial a^j}{\partial w^k}\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + a^j\frac{\partial}{\partial w^k}\left(\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i}\right)\right)\\
&= 2\frac{\partial L}{\partial w_k}\frac{\partial a^j}{\partial w^k}\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + 2\frac{\partial L}{\partial w_k}a^j\frac{\partial}{\partial w^k}\left(\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i}\right)
\end{aligned} \tag{C.50}$$

Assume that $\boldsymbol{w}$ is an optimal point in S. By Lemma C.4.2, $\boldsymbol{w}$ is also an optimal point in $\mathfrak{X}_\perp$. By Lemma C.4 in [157], $\frac{\partial L}{\partial \boldsymbol{w}}$ is an eigenvector of $\frac{\partial^2 L}{\partial w_i \partial w^j}$. Therefore, $\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i} = \lambda\frac{\partial L}{\partial w^j}$ for some

$\lambda \in \mathbb{C}$. Additionally, $a^j = M^j_\alpha \frac{\partial L}{\partial w_\alpha}$ and $\frac{\partial a^j}{\partial w^k} = M^j_\alpha \frac{\partial^2 L}{\partial w_\alpha \partial w^k}$. We are now ready to simplify both terms in equation C.50.

For the first term in equation C.50,

$$\begin{aligned}
\frac{\partial L}{\partial w_k}\frac{\partial a^j}{\partial w^k}\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i} &= \frac{\partial L}{\partial w_k} M^j_\alpha \frac{\partial^2 L}{\partial w_\alpha \partial w^k}\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i} \\
&= M^j_\alpha \left(\frac{\partial^2 L}{\partial w_\alpha \partial w^k}\frac{\partial L}{\partial w_k}\right)\left(\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i}\right) \\
&= M^j_\alpha \left(\lambda_1 \frac{\partial L}{\partial w_\alpha}\right)\left(\lambda_2 \frac{\partial L}{\partial w^j}\right) \\
&= \lambda_1 \lambda_2 M^j_\alpha \frac{\partial L}{\partial w_\alpha}\frac{\partial L}{\partial w^j} \\
&= 0
\end{aligned} \tag{C.51}$$

The last equality holds because M is anti-symmetric.

For the second term in equation C.50,

$$\begin{aligned}
\frac{\partial L}{\partial w_k} a^j \frac{\partial}{\partial w^k}\left(\frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial L}{\partial w^i}\right) &= \frac{\partial L}{\partial w_k} a^j \left(\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + \frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial^2 L}{\partial w^k \partial w^i}\right) \\
&= \frac{\partial L}{\partial w_k} M^j_\alpha \frac{\partial L}{\partial w_\alpha}\left(\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + \frac{\partial^2 L}{\partial w_i \partial w^j}\frac{\partial^2 L}{\partial w^k \partial w^i}\right) \\
&= M^j_\alpha \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + \lambda_1 \lambda_2 M^j_\alpha \frac{\partial L}{\partial w_\alpha}\frac{\partial L}{\partial w^j} \\
&= M^j_\alpha \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i}
\end{aligned} \tag{C.52}$$

In summary,

$$R[A,R]L = 2M^j_\alpha \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i}. \tag{C.53}$$

Since we assumed that $[A,R]L(\boldsymbol{w}) = 0$, when $R[A,R]L(\boldsymbol{w}) = 0$ for all $A \in S$, the gradient flow starting at an optimal point in S is optimal in S. □

Proposition C.4.3. *If* $\frac{\partial^3 L}{\partial w^k \partial w^i \partial w^j}\frac{\partial L}{\partial w^\alpha} = \frac{\partial^3 L}{\partial w^k \partial w^i \partial w^\alpha}\frac{\partial L}{\partial w^j}$ *holds for all* i,k,j,α*, then for all anti-*

symmetric matrices $M \in \mathbb{R}^{n\times n}$, $M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} = 0$.

Proof. If $\frac{\partial^3 L}{\partial w^k \partial w^i \partial w^j}\frac{\partial L}{\partial w^\alpha} = \frac{\partial^3 L}{\partial w^k \partial w^i \partial w^\alpha}\frac{\partial L}{\partial w^j}$ for all i,k,j,α, then

$$
\begin{aligned}
& M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} \\
&= \sum_{i,k,\alpha<j} M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + \sum_{i,k,\alpha>j} M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} \\
&= \sum_{i,k,\alpha<j} M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + \sum_{i,k,j>\alpha} M_j^\alpha \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_j}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^\alpha}\frac{\partial L}{\partial w^i} \\
&= \sum_{i,k,\alpha<j} M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j}\frac{\partial L}{\partial w^i} + \sum_{i,k,j>\alpha} -M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w_j}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^\alpha}\frac{\partial L}{\partial w^i} \\
&= \sum_{i,k,\alpha<j} M_\alpha^j \frac{\partial L}{\partial w_k}\frac{\partial L}{\partial w^i}\left(\frac{\partial L}{\partial w_\alpha}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^j} - \frac{\partial L}{\partial w_j}\frac{\partial^3 L}{\partial w^k \partial w_i \partial w^\alpha}\right) \\
&= 0, \qquad \text{(C.54)}
\end{aligned}
$$

where the first equality uses that the diagonal of an anti-symmetric matrix is 0, the second equality swaps α and j in the second term, the third equality uses that M is anti-symmetric. □

Appendix D

Omitted Proofs in Chapter 6

Corollary 6.4.2. *Consider a network parameter space* $\mathsf{Param}(m,h,n) = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$ *and data space* $\mathscr{D}(n,k) = \mathbb{R}^{n\times k}$. *Let* $\sigma\colon \mathbb{R}^{h\times k} \to \mathbb{R}^{h\times k}$ *be a row-wise function. Consider a function* $L_{mnhk}\colon \mathsf{Param}(m,h,n) \times \mathscr{D}(n,k) \to \mathbb{R}^{m\times k}$, *defined as* $L_{mnhk}((U,V),X) = U\sigma(VX)$ *for* $U \in \mathbb{R}^{m\times h}$, $V \in \mathbb{R}^{h\times n}$, *and* $X \in \mathbb{R}^{n\times k}$. *If there is a G-symmetry of* L_{mnhk}, *then there is a G-symmetry of* $L_{mnh'k}$ *with any* $h' > h$.

Proof. The function $L_{mnh'k}$ can be decomposed into

$$\begin{aligned} U(\sigma(VX))_{ik} &= U_{ij}\sigma(VX)_{jk} \\ &= \sum_{j=1}^{h}\sum_{l=1}^{n} U_{ij}\sigma(V_{jl}X_{lk}) \\ &= \sum_{j=1}^{h}\sum_{l=1}^{n} U_{ij}\sigma(V_{jl}X_{lk}) + \sum_{j=h+1}^{h'}\sum_{l=1}^{n} U_{ij}\sigma(V_{jl}X_{lk}) \end{aligned} \tag{D.1}$$

Note that for all i,k, the first term depends only on the first h columns of U and first h rows of V, and the second terms depends only on the rest of the columns and rows of U and V. Denoting the first h columns of U as $U_{1:h}$, the rest of the columns of U as $U_{h+1:h'}$, the first h rows of V as $V_{1:h}$, and the rest of the rows of V as $V_{h+1:h'}$, we have

$$L_{mnh'k}((U,V),X) = L_{mnhk}((U_{1:h},V_{1:h}),X) + L_{mn(h'-h)k}((U_{h+1:h'},V_{h+1:h'}),X). \tag{D.2}$$

Let $\mathsf{Param}_1 = \mathbb{R}^{m\times h} \times \mathbb{R}^{h\times n}$ and $\mathsf{Param}_2 = \mathbb{R}^{m\times(h'-h)} \times \mathbb{R}^{(h'-h)\times n}$. Then $\mathsf{Param}(m,h',n)$ $= \mathsf{Param}_1 \times \mathsf{Param}_2$. Let $S = (\mathbb{R}^{m\times k} \times \mathscr{D}^d)$ and $T = \mathbb{R}^{m\times k} \times \mathbb{R}^{m\times k}$. Define the following three functions

$$\begin{aligned} h\colon & \mathsf{Param}_1 \times \mathscr{D}^d \to (\mathbb{R}^{m\times k} \times \mathscr{D}^d) \\ f\colon & \Theta_2 \times (\mathbb{R}^{m\times k} \times \mathscr{D}^d) \to \mathbb{R}^{m\times k} \times \mathbb{R}^{m\times k} \\ j\colon & (\Theta_1 \times (\mathbb{R}^{m\times k} \times \mathbb{R}^{m\times k})) \times \mathscr{D}^d \to \mathbb{R}^{m\times k} \end{aligned} \tag{D.3}$$

by

$$\begin{aligned} h((U_{1:h},V_{1:h}),X) &= (L_{mnhk}((U_{1:h},V_{1:h}),X),X) \\ f((U_{h+1:h'},V_{h+1:h'}),(Y,X)) &= \big(L_{mn(h'-h)k}((U_{h+1:h'},V_{h+1:h'}),X),Y\big) \\ j(\big((U_{1:h},V_{1:h}),(Y',Y)\big),X) &= Y'+Y. \end{aligned} \tag{D.4}$$

Then $L_{mnh'k}(\theta,X) = j\big((\theta_1, f\big(\theta_2, h(\theta_1,X)\big)),X\big)$ for all $\theta = (\theta_1,\theta_2) \in \mathsf{Param}$ and $X \in \mathscr{D}^d$. Since L_{mnhk} has a symmetry, f has the same symmetry. By Proposition 6.4.1, $L_{mnh'k}$ also has the same symmetry. □

Corollary 6.4.3. *Let* $\mathsf{Param} = \mathsf{Param}_1 \times ... \times \mathsf{Param}_l$ *be a parameter space. Consider a list of spaces* $V_0 = \mathscr{D}^d$, $V_l = \mathbb{R}^d$, *and* V_1, ..., V_{l-1}. *Let* $L\colon \mathsf{Param} \times \mathscr{D}^d \to \mathbb{R}^d$ *be a function defined recursively by* $\{L_i\}_{i=1}^l$ *with* $L_i\colon \Theta_i \times V_{i-1} \to V_i$, *such that* $L = \phi_l$ *where* $\phi_i = L_i(\theta_i,\phi_{i-1}) \in V_i$ *and* $\phi_0 = X$. *If for some* $1 \le i \le l$, L_i *has a G-symmetry, then L has a G-symmetry.*

Proof. Define functions

$$\begin{aligned} h\colon & (\mathsf{Param}_1 \times ... \times \mathsf{Param}_{i-1} \times \mathsf{Param}_{i+1} \times ... \times \mathsf{Param}_l) \times \mathscr{D}^d \to V_{i-1} \\ f\colon & \mathsf{Param}_i \times V_{i-1} \to V_i \\ j\colon & (\mathsf{Param}_1 \times ... \times \mathsf{Param}_{i-1} \times \mathsf{Param}_{i+1} \times ... \times \mathsf{Param}_l) \times V_i \times \mathscr{D}^d \to \mathbb{R}^d \end{aligned} \tag{D.5}$$

by

$$h((\theta_1,...,\theta_{i-1},\theta_{i+1},...,\theta_l),X) = L_{i-1}(\theta_{i-1},X), \quad \text{computed using } (\theta_1,...,\theta_{i-1})$$

$$f(\theta_i,\phi_{i-1}) = L_i(\theta_i,\phi_{i-1})$$

$$j((\theta_1,...,\theta_{i-1},\theta_{i+1},...,\theta_l),\phi_i,X) = L_l(\theta_l,X), \quad \text{computed using } (\theta_l,...,\theta_{i+1}) \text{ and } \phi_i. \quad \text{(D.6)}$$

Then $L((\theta_1,...,\theta_l),X) = j\big((\theta_1,...,\theta_{i-1},\theta_{i+1},...,\theta_l), f(\theta_i, h((\theta_1,...,\theta_{i-1},\theta_{i+1},...,\theta_l), X)),X\big)$ for all $\theta = (\theta_1,\theta_2) \in \mathsf{Param}$ and $X \in \mathscr{D}^d$. By Proposition 6.4.1, if $f = L_i$ has a G-symmetry, L also has a G-symmetry. □

Bibliography

[1] Linara Adilova, Maksym Andriushchenko, Michael Kamp, Asja Fischer, and Martin Jaggi. Layer-wise linear mode connectivity. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Mohammed Adnan, Rohan Jain, Ekansh Sharma, Rahul Krishnan, and Yani Ioannou. Sparse training from random initialization: Aligning lottery ticket masks using weight symmetry. In *Forty-second International Conference on Machine Learning*, 2025.

[3] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023.

[4] Francesca Albertini, Paolo Dai Pra, et al. Recurrent neural networks: Identification and other system theoretic properties. *Neural Network Systems Techniques and Applications*, 3:1–41, 1995.

[5] Osmar Aléssio. Formulas for second curvature, third curvature, normal curvature, first geodesic curvature and first geodesic torsion of implicit curve in n-dimensions. *Computer Aided Geometric Design*, 29(3-4):189–201, 2012.

[6] Gul Sena Altintas, Gregor Bachmann, Lorenzo Noci, and Thomas Hofmann. Disentangling linear mode-connectivity. *arXiv preprint arXiv:2312.09832*, 2023.

[7] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[8] Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. *International Conference on Machine Learning*, 2023.

[9] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 29, 2016.

[10] Marco Armenta and Pierre-Marc Jodoin. The representation theory of neural networks. *Mathematics*, 9(24), 2021.

[11] Marco Armenta, Thierry Judge, Nathan Painchaud, Youssef Skandarani, Carl Lemaire, Gabriel Gibeau Sanchez, Philippe Spino, and Pierre-Marc Jodoin. Neural teleportation. *Mathematics*, 11(2):480, 2023.

[12] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *International Conference on Learning Representations*, 2018.

[13] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

[14] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[15] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[16] Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754*, 2015.

[17] Robert Bamler and Stephan Mandt. Improving optimization for models with continuous symmetry breaking. In *International Conference on Machine Learning*, pages 423–432. PMLR, 2018.

[18] Joshua Bassey, Lijun Qian, and Xianfang Li. A survey of complex-valued neural networks. *arXiv preprint arXiv:2101.12249*, 2021.

[19] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[20] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew G Wilson. Learning invariances in neural networks from training data. *Advances in neural information processing systems*, 33:17605–17616, 2020.

[21] Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, pages 769–779. PMLR, 2021.

[22] Frederik Benzing, Simon Schug, Robert Meier, Johannes Von Oswald, Yassir Akram,

Nicolas Zucchet, Laurence Aitchison, and Angelika Steger. Random initialisations performing above chance and how to find them. *14th Annual Workshop on Optimization for Machine Learning (OPT2022)*, 2022.

[23] Alan J Bray and David S Dean. Statistics of critical points of gaussian fields on large-dimensional spaces. *Physical review letters*, 98(15):150201, 2007.

[24] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

[25] John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

[26] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[27] David Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. *ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM)*, RSRE-MEMO-4148, 03 1988.

[28] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *International Conference on Learning Representations*, 2017.

[29] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

[30] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[31] Yaim Cooper. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.

[32] Marvin F da Silva, Felix Dangel, and Sageev Oore. Hide & seek: Transformer symmetries obscure sharpness & riemannian geometry finds it. In *Forty-second International Conference on Machine Learning*, 2025.

[33] Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with lie algebra convolutional network. *Advances in Neural Informa-*

tion Processing Systems, 34:2503–2515, 2021.

[34] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[35] Lijun Ding, Dmitriy Drusvyatskiy, Maryam Fazel, and Zaid Harchaoui. Flat minima generalize for low-rank matrix recovery. *arXiv preprint arXiv:2203.03756*, 2022.

[36] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[37] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[38] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Neural Information Processing Systems*, 2018.

[39] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[40] Omer Elkabetz and Nadav Cohen. Continuous vs. discrete optimization of deep neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[41] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *International Conference on Learning Representations*, 2022.

[42] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[43] Damien Ferbach, Baptiste Goujaud, Gauthier Gidel, and Aymeric Dieuleveut. Proving linear mode connectivity of neural networks via optimal transport. *arXiv preprint arXiv:2310.19103*, 2023.

[44] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[45] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International*

conference on machine learning, pages 3318–3328. PMLR, 2021.

[46] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

[47] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[48] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[49] K. Fukumizu and S. Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000.

[50] Alex Gabel, Victoria Klein, Riccardo Valperga, Jeroen SW Lamb, Kevin Webster, Rick Quax, and Efstratios Gavves. Learning lie group symmetry transformations with neural networks. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 50–59. PMLR, 2023.

[51] Iordan Ganev, Twan van Laarhoven, and Robin Walters. Universal approximation and model compression for radial neural networks. *arXiv preprint arXiv:2107.02550v2*, 2022.

[52] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[53] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[54] Grzegorz Głuch and Rüdiger Urbanke. Noether: The more things change, the more stay the same. *arXiv preprint arXiv:2104.05508*, 2021.

[55] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. *Advances in Neural Information Processing Systems*, 2022.

[56] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Using mode connectivity for loss landscape analysis. *arXiv preprint arXiv:1806.06977*, 2018.

[57] Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of relu networks. In *International Conference on Machine Learning*, pages 11734–11760. PMLR, 2023.

[58] J Elisenda Grigsby, Kathryn Lindsey, Robert Meyerhoff, and Chenxi Wu. Functional dimension of feedforward relu neural networks. *Advances in Mathematics*, 482:110636, 2025.

[59] Nate Gruver, Marc Anton Finzi, Micah Goldblum, and Andrew Gordon Wilson. The lie derivative for measuring learned equivariance. In *The Eleventh International Conference on Learning Representations*, 2022.

[60] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *35th International Conference on Machine Learning*, 2018.

[61] Elad Hazan. Lecture notes: Optimization for machine learning. *arXiv preprint arXiv:1909.03550*, 2019.

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

[63] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[65] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[66] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[67] Lei Huang, Xianglong Liu, Bo Lang, Adams Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[68] Dongsung Huh. Curvature-corrected learning dynamics in deep neural networks. In *International Conference on Machine Learning*, pages 4552–4560. PMLR, 2020.

[69] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. pmlr, 2015.

[70] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *Conference on Uncertainty in Artificial Intelligence*, 2018.

[71] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[72] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair. *International Conference on Learning Representations*, 2023.

[73] Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. *International Conference on Learning Representations*, 2023.

[74] Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. *Advances in neural information processing systems*, 37:106800–106840, 2024.

[75] Pavan Karjol, Rohan Kashyap, Aditya Gopalan, and A. P. Prathosh. A unified framework for discovering discrete symmetries. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 793–801. PMLR, 2024.

[76] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*, 2017.

[77] Minyoung Kim, Da Li, Shell X Hu, and Timothy Hospedales. Fisher sam: Information geometry and sharpness aware minimisation. In *International Conference on Machine Learning*, pages 11148–11161. PMLR, 2022.

[78] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[79] Sven Krippendorf and Marc Syvaeri. Detecting symmetries with neural networks. *Machine Learning: Science and Technology*, 2(1):015010, 2020.

[80] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny

images. Technical report, University of Toronto, 2009.

[81] Rohith Kuditipudi, Xiang Wang, Holden Lee, Yi Zhang, Zhiyuan Li, Wei Hu, Rong Ge, and Sanjeev Arora. Explaining landscape connectivity of low-cost solutions for multilayer nets. *Advances in neural information processing systems*, 32, 2019.

[82] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel LK Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. In *International Conference on Learning Representations*, 2021.

[83] Věra Kůrková and Roman Neruda. Uniqueness of functional representations by gaussian basis function networks. In *ICANN'94: Proceedings of the International Conference on Artificial Neural Networks Sorrento, Italy, 26–29 May 1994 Volume 1, Parts 1 and 2 4*, pages 471–474. Springer, 1994.

[84] Henry Kvinge, Tegan Emerson, Grayson Jorgenson, Scott Vasquez, Tim Doster, and Jesse Lew. In what ways are deep neural networks invariant and how should we measure this? *Advances in Neural Information Processing Systems*, 35:32816–32829, 2022.

[85] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914. PMLR, 2021.

[86] Lucas Laird, Bo Zhao, Rose Yu, and Robin Walters. Data-free transformer quantization using parameter-space symmetry. *Workshop on High-dimensional Learning Dynamics (HiLD)*, 2025.

[87] Serge Lang. *Algebra*. Graduate Texts in Mathematics. Springer, 2002.

[88] Olivier Laurent, Emanuel Aldea, and Gianni Franchi. A symmetry-aware exploration of bayesian neural network posteriors. In *The Twelfth International Conference on Learning Representations*, 2024.

[89] John Lee. *Introduction to Topological Manifolds*, volume 202. Springer Science & Business Media, 2010.

[90] John M Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics, vol 218. Springer, New York, NY, 2013.

[91] Daniel Lengyel, Janith Petangoda, Isak Falk, Kate Highnam, Michalis Lazarou, Arinbjörn Kolbeinsson, Marc Peter Deisenroth, and Nicholas R Jennings. Genni: Visualising the geometry of equivalences for neural network identifiability. *arXiv preprint arXiv:2011.07407*, 2020.

[92] Zhiyuan Li, Srinadh Bhojanapalli, Manzil Zaheer, Sashank Reddi, and Sanjiv Kumar. Robust training of neural networks using scale invariant architectures. In *International Conference on Machine Learning*, pages 12656–12684. PMLR, 2022.

[93] Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In *The Twelfth International Conference on Learning Representations*, 2024.

[94] Derek Lim, Theo Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The empirical impact of neural parameter symmetries, or lack thereof. *Advances in Neural Information Processing Systems*, 37:28322–28358, 2024.

[95] Ekdeep Singh Lubana, Eric J Bigelow, Robert P Dick, David Krueger, and Hidenori Tanaka. Mechanistic mode connectivity. In *International Conference on Machine Learning*, pages 22965–23004. PMLR, 2023.

[96] James Martens and Roger B Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on International Conference on Machine Learning*. PMLR, 2018.

[97] Flavio Martinelli, Berfin Simsek, Wulfram Gerstner, and Johanni Brea. Expand-and-cluster: Parameter recovery of neural networks. In *International Conference on Machine Learning*, pages 34895–34919. PMLR, 2024.

[98] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different: Recovering neural network quantization error through weight factorization. In *International Conference on Machine Learning*, pages 4486–4495. PMLR, 2019.

[99] Qi Meng, Shuxin Zheng, Huishuai Zhang, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. $\mathscr{G}$-SGD: Optimizing relu neural networks in its positively scale-invariant space. *International Conference on Learning Representations*, 2019.

[100] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[101] Hancheng Min, Salma Tarmoun, René Vidal, and Enrique Mallada. On the explicit role of initialization on the convergence and implicit bias of overparametrized linear networks. In *International Conference on Machine Learning*. PMLR, 2021.

[102] Aaron Mishkin, Alberto Bietti, and Robert M. Gower. Level set teleportation: the good, the bad, and the ugly. In *OPT2023: 15th Annual Workshop on Optimization for Machine Learning*, 2023.

[103] Artem Moskalev, Anna Sepliarskaia, Erik J Bekkers, and Arnold WM Smeulders. On genuine invariance learning without weight-tying. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 218–227. PMLR, 2023.

[104] Artem Moskalev, Anna Sepliarskaia, Ivan Sosnovik, and Arnold Smeulders. Liegg: Studying learned lie group generators. *Advances in Neural Information Processing Systems*, 35:25212–25223, 2022.

[105] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019.

[106] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 25790–25816. PMLR, 2023.

[107] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-SGD: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[108] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

[109] Quynh Nguyen. On connected sublevel sets in deep learning. In *International Conference on Machine Learning*, pages 4790–4799. PMLR, 2019.

[110] Quynh Nguyen. A note on connectivity of sublevel sets in deep learning. *arXiv preprint arXiv:2101.08576*, 2021.

[111] Quynh N Nguyen, Pierre Bréchet, and Marco Mondelli. When are solutions connected in deep networks? *Advances in Neural Information Processing Systems*, 34:20956–20969, 2021.

[112] Emmy Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, page 235–257, 1918.

[113] Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. *35th Conference on Neural Information Processing Systems*, 2021.

[114] Henning Petzka, Martin Trimmel, and Cristian Sminchisescu. Notes on the symmetries of 2-layer relu-networks. In *Proceedings of the Northern Lights Deep Learning Workshop*, volume 1, pages 6–6, 2020.

[115] Fabrizio Pittorino, Antonio Ferraro, Gabriele Perugini, Christoph Feinauer, Carlo Baldassi, and Riccardo Zecchina. Deep networks on toroids: Removing symmetries reveals the structure of flat regions in the landscape geometry. In *Proceedings of the 39th International Conference on Machine Learning*, pages 17759–17781, 2022.

[116] Vasco Portilheiro. Quantifying lie group learning with local symmetry error. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2023.

[117] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

[118] Theo Putterman, Derek Lim, Yoav Gelberg, Stefanie Jegelka, and Haggai Maron. Learning on loras: Gl-equivariant processing of low-rank weight spaces for large finetuned models. *arXiv preprint arXiv:2410.04207*, 2024.

[119] David W Romero and Suhas Lohit. Learning partial equivariances from data. *Advances in Neural Information Processing Systems*, 35:36466–36478, 2022.

[120] HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[121] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[122] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[123] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*, 2014.

[124] Ben Shaw, Abram Magner, and Kevin R Moon. Symmetry discovery beyond affine transformations. *Advances in Neural Information Processing Systems*, 37:112889–112918, 2024.

[125] Aleksandr Mikhailovich Shelekhov. On the curvatures of a curve in n-dimensional euclidean space. *Russian Mathematics*, 65(11):46–58, 2021.

[126] Guohao Shen. Complexity of feed-forward neural networks from the perspective of functional equivalence. In *International Conference on Machine Learning*. PMLR, 2024.

[127] Alexander Shevchenko and Marco Mondelli. Landscape connectivity and dropout stability of sgd solutions for over-parameterized neural networks. In *International Conference on Machine Learning*, pages 8773–8784. PMLR, 2020.

[128] Berfin Şimşek, François Ged, Arthur Jacot, Francesco Spadaro, Clément Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pages 9722–9732. PMLR, 2021.

[129] Sidak Pal Singh, Linara Adilova, Michael Kamp, Asja Fischer, Bernhard Schölkopf, and Thomas Hofmann. Landscaping linear mode connectivity. *arXiv preprint arXiv:2406.16300*, 2024.

[130] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[131] Sho Sonoda, Hideyuki Ishi, Isao Ishikawa, and Masahiro Ikeda. Joint group invariant functions on data-parameter domain induce universal neural networks. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2023.

[132] Gustav Sourek, Filip Zelezny, and Ondrej Kuzelka. Lossless compression of structured convolutional models via lifting. In *International Conference on Learning Representations*, 2021.

[133] Sebastian U. Stich. Unified optimal analysis of the (stochastic) gradient method. *CoRR*, 2019.

[134] Hidenori Tanaka and Daniel Kunin. Noether's learning dynamics: Role of symmetry breaking in neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[135] Salma Tarmoun, Guilherme Franca, Benjamin D Haeffele, and Rene Vidal. Understanding the dynamics of gradient flow in overparameterized linear models. In *International Conference on Machine Learning*, pages 10153–10161. PMLR, 2021.

[136] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311, 2020.

[137] Hoang V Tran, Thieu N Vo, Tho H Tran, An T Nguyen, and Tan M Nguyen. Monomial matrix group equivariant neural functional networks. *arXiv preprint arXiv:2409.11697*, 2024.

[138] Alonso Urbano and David W Romero. Self-supervised detection of perfect and partial

input-dependent symmetries. *arXiv preprint arXiv:2312.12223*, 2023.

[139] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *Advances in Neural Information Processing Systems*, 2017.

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[141] Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32, 2019.

[142] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[143] Andre Wibisono and Ashia C Wilson. On accelerated methods in optimization. *arXiv preprint arXiv:1509.03616*, 2015.

[144] Jonas Gregor Wiese, Lisa Wimmer, Theodore Papamarkou, Bernd Bischl, Stephan Günnemann, and David Rügamer. Towards efficient mcmc sampling in bayesian neural networks by exploiting symmetry. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD): Research Track*, pages 459–474, 2023.

[145] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

[146] Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

[147] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[148] Zihao Wu, Juncheng Dong, Ahmed Aloui, and Vahid Tarokh. Teleportation with null space gradient projection for optimization acceleration. *arXiv preprint arXiv:2502.11362*, 2025.

[149] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[150] Tim Xiao, Weiyang Liu, and Robert Bamler. A compact representation for bayesian

neural networks by removing permutation symmetry. *UniReps: Unifying Representations in Neural Models (NeurIPS 2023 Workshop)*, 2023.

[151] Jianke Yang, Nima Dehmamy, Robin Walters, and Rose Yu. Latent space symmetry discovery. In *International Conference on Machine Learning*, pages 56047–56070. PMLR, 2024.

[152] Jianke Yang, Robin Walters, Nima Dehmamy, and Rose Yu. Generative adversarial symmetry discovery. *International Conference on Machine Learning*, 2023.

[153] David Yunis, Kumar Kshitij Patel, Pedro Henrique Pamplona Savarese, Gal Vardi, Jonathan Frankle, Matthew Walter, Karen Livescu, and Michael Maire. On convexity and linear mode connectivity in neural networks. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.

[154] Guy Zamir, Aryan Dokania, Bo Zhao, and Rose Yu. Improving learning to optimize using parameter symmetries. *arXiv preprint arXiv:2504.15399*, 2025.

[155] Binchi Zhang, Zaiyi Zheng, Zhengzhang Chen, and Jundong Li. Beyond the permutation symmetry of transformers: The role of rotation for model fusion. *International Conference on Machine Learning*, 2025.

[156] David W Zhang, Miltiadis Kofinas, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, and Cees GM Snoek. Neural networks are graphs! graph neural networks for equivariant processing of neural networks. *The 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40th International Conference on Machine Learning*, 2023.

[157] Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Symmetry teleportation for accelerated optimization. *Advances in Neural Information Processing Systems*, 2022.

[158] Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Finding symmetry in neural network parameter spaces. In *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024.

[159] Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Understanding mode connectivity via parameter space symmetry. In *International Conference on Machine Learning*, 2025.

[160] Bo Zhao, Iordan Ganev, Robin Walters, Rose Yu, and Nima Dehmamy. Symmetries, flat minima, and the conserved quantities of gradient flow. *International Conference on Learning Representations*, 2023.

[161] Bo Zhao, Robert M Gower, Robin Walters, and Rose Yu. Improving convergence and generalization using parameter symmetries. *International Conference on Learning Repre-*

sentations, 2024.

[162] Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *Transactions on Machine Learning Research*, 2026.

[163] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. *International Conference on Learning Representations*, 2020.

[164] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. *International Conference on Learning Representations*, 2021.

[165] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in Neural Information Processing Systems*, 36, 2023.

[166] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296, 2020.

[167] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going beyond linear mode connectivity: The layerwise linear feature connectivity. *arXiv preprint arXiv:2307.08286*, 2023.

[168] Liu Ziyin. Symmetry leads to structured constraint of learning. In *International Conference on Machine Learning*. PMLR, 2024.

[169] Liu Ziyin, Ekdeep Singh Lubana, Masahito Ueda, and Hidenori Tanaka. What shapes the loss landscape of self-supervised learning? *International Conference on Learning Representations*, 2023.

[170] Liu Ziyin, Yizhou Xu, and Isaac L. Chuang. Remove symmetries to control model expressivity. In *The Thirteenth International Conference on Learning Representations*, 2025.