Journal of Machine Learning Research 26 (2025) 1-70 Submitted 12/24; Revised 5/25; Published 6/25

On the Ability of Deep Networks to Learn Symmetries from Data – A Neural Kernel Theory

Andrea Perin ANDREA.PERIN@AALTO.FI
Department of Computer Science
Aalto University
Espoo, Finland

Stéphane Deny STEPHANE.DENY.PRO@GMAIL.COM
Department of Computer Science
Department of Neuroscience and Biomedical Engineering
Aalto University
Espoo, Finland

Editor: Aapo Hyvärinen

Abstract

Symmetries (transformations by group actions) are present in many datasets, and leveraging them holds considerable promise for improving predictions in machine learning. In this work, we aim to understand when and how deep networks—with standard architectures trained in a standard, supervised way—*learn* symmetries from data. Inspired by real-world scenarios, we study a classification paradigm where data symmetries are only *partially observed* during training: some classes include all transformations of a cyclic group, while others—only a subset. We ask: under which conditions will deep networks correctly classify the partially sampled classes?

In the infinite-width limit, where neural networks behave like kernel machines, we derive a *neural kernel theory of symmetry learning.* The group-cyclic nature of the dataset allows us to analyze the Gram matrix of neural kernels in the Fourier domain; here we find a simple characterization of the generalization error as a function of class separation (signal) and class-orbit density (noise). This characterization reveals that generalization can only be successful when the local structure of the data prevails over its non-local, symmetry-induced structure, in the kernel space defined by the architecture. This occurs when (1) classes are sufficiently distinct and (2) class orbits are sufficiently dense.

We extend our theoretical treatment to any finite group, including non-abelian groups. Our framework also applies to equivariant architectures (e.g., CNNs), and recovers their success in the special case where the architecture matches the inherent symmetry of the data. Empirically, our theory reproduces the generalization failure of finite-width networks (MLP, CNN, ViT) trained on partially observed versions of rotated-MNIST. We conclude that conventional deep networks lack a mechanism to learn symmetries that have not been explicitly embedded in their architecture *a priori*. In the future, our framework could be extended to guide the design of architectures and training procedures able to learn symmetries from data.

All code is available at `https://github.com/Andrea-Perin/gpsymm`.

Keywords: deep learning, symmetry, neural kernel methods, gaussian processes, spectral methods

1. Introduction

The ability to make accurate predictions depends on how well one understands the structure of a problem. Physics, in particular, has achieved remarkable success in predicting natural phenomena by capturing their structure in compact mathematical equations. A key aspect of this structure lies in the concept of *symmetry* (Noether, 1918; Gross, 1996). Symmetries describe transformations by group actions that do not affect the identity of objects. For example, a chair remains a chair whether it is presented upright or upside down (a transformation in $SO(3)$). In the context of deep learning, nowadays widely used for prediction tasks, a crucial question is whether deep networks have mechanisms to recognize and exploit data symmetries for effective predictions. For example, can deep networks learn to predict the identity of objects independently of their viewpoint? And importantly, do they need to be exposed to *all object classes in all possible poses* during training, or is exposure to some classes in some poses sufficient for them to *capture the concept* of pose invariance?

The field of geometric deep learning develops theories and methods that enable neural networks to take advantage of problem symmetries (Bronstein et al., 2021). In particular, equivariant neural networks (Cohen and Welling, 2016) can ensure representation equivariance and classification invariance to prespecified symmetries. However, equivariant architectures require one to know the symmetry of the problem in advance. Here, we aim to understand whether conventional deep networks—which have not been explicitly designed to capture a prespecified symmetry—can *learn symmetries directly from data.*

We focus on a supervised classification paradigm where the symmetries of the data are only partially observed during training : for some classes, all possible transformations of a cyclic group are observed during training, while for other classes only a subset of transformations is observed. This scenario replicates realistic real-world learning problems. For example, a child during their development sees a few objects in all possible 3D poses (e.g., the toys they can manipulate), and many objects in only some poses (e.g., heavy furniture). *Should we expect a deep network trained on such a data diet to generalize to the partially sampled classes (e.g., recognize a piece of furniture seen from an unusual viewpoint at test time)?* In this work, we study the conditions under which deep networks correctly extrapolate the symmetry invariance to the partially sampled classes.

To characterize the generalization capabilities of deep networks in the presence of data symmetries, we rely on the equivalence between infinitely wide neural networks and kernel machines (Neal, 1996; Lee et al., 2018; Jacot et al., 2018). Typical neural kernels (MLP, CNN) greatly simplify when computed over a dataset generated by a cyclic group action (they become circulant), allowing an interpretable analysis in the Fourier domain of when and how symmetries are correctly learned. *We find that the generalization behavior of networks is predicted by a simple ratio of inverse kernel frequency powers computed over orbits of the cyclic group.* Our analysis of this formula makes clear that deep networks (as described by their kernel equivalent) are *a priori* unable to leverage data symmetries for generalization. Successful generalization is yet possible, in cases where the local structure of the data prevails over its non-local, symmetry-induced structure. This happens in particular (1) when classes are sufficiently well separated in kernel space, and (2) when the symmetric structure of the data is sufficiently local in kernel space. Importantly, while there is no guaranteed equivalence between finite-width networks and their infinite-width counterpart, our spectral kernel theory captures well the behavior of normally trained finite-width networks in all our experiments on rotated-MNIST, a version of the well-known handwritten character dataset (LeCun et al., 1998) augmented with rotations.

Outline. In Section 2, we briefly review prior theoretical work and practical methods developed for deep learning in relation to symmetries. In Section 3, we illustrate a simple symmetry learning problem, where deep networks with common architectures (MLP, CNN, ViT) are trained and evaluated on partial views of rotated-MNIST. In Section 4, we analyze theoretically how kernel machines in general—and neural networks in particular—behave on datasets presenting symmetries. We build our theory through scenarios of increasing complexity, from a simple Gaussian kernel applied to a circular dataset, to deep neural kernels with or without equivariant architectures (MLPs and CNNs) applied to an affine group transformation such as rotated-MNIST. In all cases, we show both theoretically and empirically the inadequacy of conventional neural architectures trained with supervision to learn symmetries that have not been embedded in their kernel design *a priori*. In Section 5, we discuss how our work provides theoretical tools which could be helpful in identifying and discovering architectures and training procedures able to learn symmetries from data. In Appendix C, we extend our theory to any finite group, including non-abelian groups.

2. Prior Work on Deep Learning, Symmetries, and Neural Kernels

The interplay between deep learning and symmetries has been studied extensively, both empirically and theoretically, for finite-width networks and in the infinite-width limit (where kernel analogies apply). Here we attempt a condensed review of these efforts. In the following, our operational definition of *dataset symmetries* is a set of transformations by group action that do not change the class identity of objects present in a dataset. These group transformations may act directly in the native space of the dataset (e.g., image translations) or in a latent space affecting the dataset (e.g., images of 3D-rotating objects).

Empirically, the inability of conventional neural networks to capture symmetries present in datasets has been observed in many different contexts. Conventional networks have been shown to fail to extrapolate a simple periodic function (Ziyin et al., 2020). In vision, studies have investigated the generalization capabilities of deep networks to recognize objects undergoing changes in pose (Alcorn et al., 2019; Madan et al., 2022; Abbas and Deny, 2023; Siddiqui et al., 2023), size (Ibrahim et al., 2023), mirror symmetry (Sundaram et al., 2022), lighting conditions (Madan et al., 2025), and shown a substantial degradation of network performance in these conditions. Recently, Ollikka et al. (2025) also showed that humans beat state-of-the-art deep networks and most vision-language models at recognizing objects in unusual poses. In language, compositional generalization has also been framed as capturing permutation symmetries, which traditional language models have been shown to fail at. *In our work, we study from a theoretical point of view why networks fail to generalize on data symmetries despite being exposed to them partially during training. To do so, we characterize network behavior on symmetric datasets in the kernel limit.*

Network architectures have been designed to be equivariant to prespecified symmetries (Gens and Domingos, 2014; Cohen and Welling, 2016; Worrall et al., 2017; Cohen et al., 2019a,b; Bekkers, 2021; Weiler et al., 2021), or conserve certain physical quantities (Greydanus et al., 2019; Cranmer et al., 2020; Finzi et al., 2020; van der Ouderaa et al., 2024). Network architectures have also been designed to respect the topology of a problem (for a review, see Hajij et al., 2023). Relaxed and adaptive equivariance schemes have also been proposed (Elsayed et al., 2020; Zhou et al., 2021; D'Ascoli et al., 2021; Wang et al., 2022; Yeh et al., 2022; Kaba and Ravanbakhsh, 2023; van der Ouderaa et al., 2023). The failure of traditional convolutional architectures to be fully equivariant has been observed (e.g., Azulay and Weiss, 2019), and some measure of approximate equivariance proposed (Gruver et al., 2023). Others have investigated how equivariant representations affect the capacity of group-invariant linear readouts (Farrell et al., 2022). Beyond equivariance, symmetries inherent to the architecture of deep networks have also been studied, and shown to affect their learning dynamics and solutions (Tanaka and Kunin, 2021; Simsek et al., 2021; Ainsworth et al., 2023). *In our work, we study the ability of neural networks to generalize on datasets presenting symmetries to which they have* not *been designed to be equivariant.*

The effects of training networks on symmetry-augmented versions of a dataset have also been scrutinized. Recently, the efficiency of training networks on augmented data vs. enforcing equivariance in the architecture has been carefully characterized at scale (Brehmer et al., 2024) and on a variety of datasets (Vadgama et al., 2025). Moskalev et al. (2023) clarify that training on augmented datasets does not produce genuine equivariance, in the sense that the trained networks may not be equivariant outside the training distribution. Recent theoretical results show, however, that training on a perfectly augmented dataset produces emergent equivariant representations in *ensembles* of networks, in and outside the training set, both in the finite (Nordenfors and Flinth, 2024) and infinite-width limit (Gerken and Kessel, 2024). *In our work we ask a related but different question: we study the generalization capabilities of networks trained on a dataset containing unknown symmetries—where no systematic augmentation scheme or equivariant architecture has been enforced. The symmetries in the training dataset may be well represented for some classes, but not others. The question we ask is then: how well will networks generalize the symmetry invariance to the partially sampled classes?*

Beyond equivariant and augmentation methods to deal with prespecified symmetries, approaches to learning the symmetries present in data have been proposed (Culpepper and Olshausen, 2009; Jaderberg et al., 2015; Sohl-Dickstein et al., 2017; Dupont et al., 2020; Benton et al., 2020; Connor and Rozell, 2020; Zhou et al., 2021; Keller and Welling, 2021b; Pérez Rey et al., 2023; Sanborn et al., 2023; Connor et al., 2024; van der Linden et al., 2024; Yang et al., 2024b) **and studied theoretically** (Anselmi et al., 2019; Pfau et al., 2020; Anselmi et al., 2023; Marchetti et al., 2024). Self-supervised learning approaches build invariance (Zemel and Hinton, 1990; Chen et al., 2020; Zbontar et al., 2021; Ibrahim et al., 2022) or equivariance (Garrido et al., 2024) to predefined symmetries by feeding augmentations of the dataset to two identical versions of the network and matching their latent representations. Disentanglement has also been framed as the problem of learning symmetries from data (Higgins et al., 2018; Mercatali et al., 2022). Subsequent work has shown, however, that topological defects result from attempting to learn disentangled representations of even the simplest symmetries such as affine transformations (Bouchacourt et al., 2021; Esmaeili et al., 2023). *In our work, we focus on characterizing the symmetry learning ability of conventional architectures (MLP, CNN) trained with traditional supervised learning. However, our theoretical framework could potentially be extended to characterize the ability of these other methods to capture the symmetries of data.*

Beyond symmetries, the interplay between machine learning and data geometry has been studied for many types of data structure: compositional (Sabour et al., 2017; Schott et al., 2022; Liang et al., 2024; Wiedemer et al., 2023; Lippl and Stachenfeld, 2025; Yang et al., 2025)**, hierarchical** (Saxe et al., 2019; Mel and Ganguli, 2021)**, data lying on a manifold** (Goldt et al., 2020; Gerace et al., 2022) **or on separate manifolds** (Chung et al., 2018; Cohen et al., 2020; Sorscher et al., 2022). *Our work focuses on datasets presenting symmetries (i.e., group-action induced structures), an important type of structure found in the physical world and present in many datasets.*

Two main kernel theories of deep learning exist, which identify deep networks with kernel machines (Neal, 1996; Jacot et al., 2018).[1] Both theories apply in the limit where the networks have infinite width (infinite number of neurons within a layer or infinite number of channels in convolutional networks), and both identify training a network with doing simple Gaussian Process regression over a fixed kernel determined by the network architecture. The Neural Network Gaussian Process (NNGP) theory (Neal, 1996) characterizes the distribution of functions produced by random draws of the network parameters (its weights and biases), assuming a distribution from which the parameters are drawn (typically i.i.d. Normal). Conditioned on a training set, the NNGP characterizes the distribution of solutions (i.e., functions passing through the training input-label pairs) under the assumed distribution of parameters, and can be used to make predictions on a testing set. The Neural Tangent Kernel theory (NTK) (Jacot et al., 2018) describes the distribution of solutions obtained by training all layers with gradient descent, from random initial conditions. It is allegedly the most realistic theory (although see Avidan et al. (2025) for a more nuanced account) and the one we use in this work. Nonetheless, we repeated our analyses using the NNGP kernel instead of the NTK and it did not change our conclusions (not shown). Neural kernel theories assume networks to have infinite width, which is a big approximation in practice. For example, Fort et al. (2020) show that the kernel of finite-width networks changes rapidly during the first few epochs of training to a more favourable kernel for the task, breaking the frozen kernel assumption which is only provable in the infinite-width limit. *However, in all our experiments, we find that kernel theories adequately capture the generalization behavior of finite-width, normally trained networks on datasets presenting symmetries.*

Finally, in an insightful line of work, Bordelon et al. (2020); Canatar et al. (2021) **relate generalization properties of infinite-width deep networks to spectral properties of their kernel**. They do not, however, explore the specific interplay between spectral kernel properties and data symmetries. *In our work, we find that the symmetric nature*

[1] We refer the reader unfamiliar with neural kernel theories to the excellent lecture notes of Adityanarayanan Radhakrishnan available online on this topic (`https://aditradha.com/lecture-notes/`). We also provide an intuitive explanation of neural kernel theories in Appendix A.

of the dataset considerably simplifies the spectral description of neural kernels, allowing better interpretability of the factors on which generalization depends.

We next propose a simple empirical study to illustrate the problem of symmetry learning and highlight the failure of a range of common architectures on this problem.

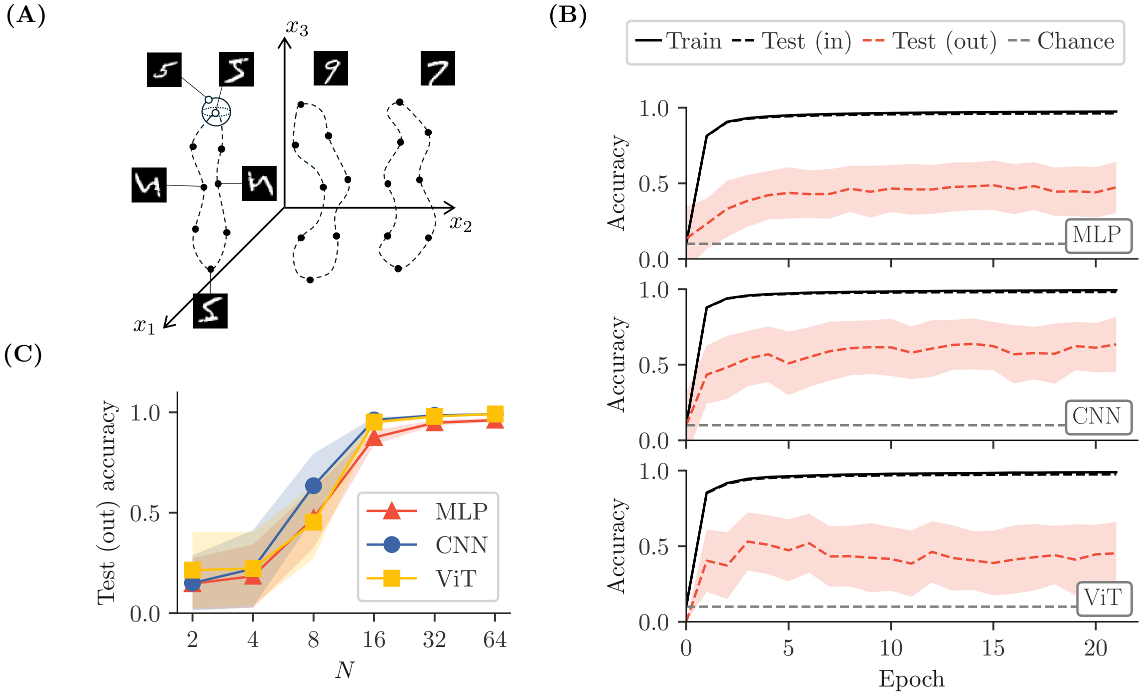


Figure 1: **Common deep network architectures fail to extrapolate symmetries from a partially observed version of rotated-MNIST. A**: A conceptual sketch of the learning task. For samples in the leave-out class (digit "5" in this example), the upright pose is not included in the training set (empty circles). For samples in the remaining classes, all poses are included in the training set (full circles). A model that can generalize rotations should classify the missing upright samples correctly. **B**: The accuracies of the three models tested respectively on the training set, *in-test* set (normal i.i.d. test set) and *out-test* set (i.e., left-out pose of the leave-out class). Chance levels (10%) are also reported as a baseline. Error shades represent 95% confidence intervals computed over all tested leave-out classes (n = 10). **C**: Out-test accuracy for the three models as a function of the number of angles in the rotation orbit. Error shades as in *B*.

3. Illustration of a Symmetry Learning Problem

To illustrate the problem of symmetry learning, we train a range of networks with different architectures (MLP, ConvNet, ViT-S (Dosovitskiy et al., 2021)) on partially observed versions of rotated-MNIST (details of the architectures in App. D).

We start by constructing the rotated-MNIST dataset: for all samples in MNIST, we generate N rotated samples, where the rotation angles are multiples of $2\pi/N$. We choose a digit class, which we refer to as the *leave-out class* (Fig. 1A). We define the "training" and the "in-test" splits as a 90/10 random split of the whole rotated MNIST dataset, *except* the upright samples of the leave-out class. These samples, instead, constitute what we call the "out-test" split.[2]

We report the accuracies of the three models in the case of 8 rotation angles in Fig. 1B. The in-test accuracy closely tracks the training set accuracy; however, the out-test accuracy is considerably lower for all the considered models. Training for very long times (>1000 epochs) did not improve out-test accuracy (no grokking happens).

We find that increasing the number of angles leads to better out-test performance across models (Fig. 1C). Successful generalization is achieved when the number of angles sampled is sufficiently large.

In the following, we propose a theory to understand the generalization abilities of deep networks on problems involving learning from partially observed symmetries. Fully developed, our theory gives a precise account of the generalization behavior of conventionally trained networks on rotated-MNIST (Section 4.5 and Fig. 6).

4. A Neural Kernel Theory of Symmetry Learning

Essentially, the argument of this paper relies on the two following observations: (1) in the Neural Tangent Kernel (NTK) limit, training deep networks is analog to performing kernel regression; (2) kernel regression has a greatly simplified expression on datasets presenting symmetries, allowing a simple geometric interpretation of the factors leading to generalization (or lack thereof). We provide in Appendix A a brief and intuitive description of NTK. We present below the main theoretical result of the paper, which consists in the simplified expression for kernel regression on a symmetric dataset.

4.1 General Definitions

We start by recalling basic definitions relating to kernel methods.

Definition 1 (Kernel Function) *A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ takes two inputs $x, x' \in \mathcal{X}$ and returns a scalar value representing their similarity. It can be defined as the inner product between the images of the inputs under a certain feature map $\varphi : \mathcal{X} \to \mathcal{H}$:*

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

Where:

[2] Note that predicting the labels of the out-test split is an out-of-distribution (OOD) generalization task. In this work, we study whether networks are able to perform this OOD task in virtue of the symmetric nature of the dataset.

- *$\varphi(x)$ is a mapping from the original input space $\mathcal{X}$ to a higher-dimensional feature space $\mathcal{H}$ (possibly infinite-dimensional).*
- *$\langle \cdot, \cdot \rangle$ represents the inner product in $\mathcal{H}$.*

Knowing the kernel function allows one to skip explicit computation of the mapping $\varphi(x)$ and directly obtain the inner product.

Definition 2 (Gram Matrix) *Given a dataset $\{x_1, x_2, \ldots, x_N\}$ with N data points, and a kernel function $k(x, x')$, the* Gram matrix *K is defined as the matrix of pairwise kernel evaluations between all pairs of data points:*

$$K_{ij} = k(x_i, x_j)$$

This matrix captures the pairwise relationships between all data points in the feature space induced by the kernel.

Definition 3 (Circulant Gram Matrix) *A Gram matrix K is said to have a circulant structure if each row of the matrix is a cyclic shift of the previous row. Specifically, the matrix K is circulant if its entries satisfy the following condition:*

$$K_{ij} = c_{(i-j) \mod N}$$

where N is the number of data points (or the size of the matrix), and c_k represents the entries of a vector $\mathbf{c} = \{c_0, c_1, \ldots, c_{N-1}\}$, which is periodic with period N. This means the matrix K can be written in the form:

$$K = \begin{bmatrix} c_0 & c_1 & c_2 & \cdots & c_{N-1} \\ c_{N-1} & c_0 & c_1 & \cdots & c_{N-2} \\ c_{N-2} & c_{N-1} & c_0 & \cdots & c_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \cdots & c_0 \end{bmatrix}$$

In other words, the matrix entries are determined by a single vector $\mathbf{c}$, and each row is a shifted version of this vector, where the shifts are cyclic (i.e., they wrap around the matrix).

4.2 Spectral Error

When the Gram matrix is circulant over a dataset made of two interleaved classes, we can derive a simplified formula for kernel regression.

Proposition 4 *Consider a kernel function and its Gram matrix on a dataset. The dataset has an even number of points $2N$, where the data points are from two classes ordered in an interleaved manner with labels $+1$ and -1. Assume that the Gram matrix is circulant on this ordered dataset. Recalling that a circulant matrix is diagonalized by the* Discrete Fourier Transform *(DFT), kernel regression on a missing point from this dataset will result in the following error, referred to as the* spectral error*:*

$$\varepsilon_s = \frac{\lambda_N^{-1}}{\langle \lambda^{-1} \rangle}, \tag{1}$$

where λ_N denotes the eigenvalue of the largest frequency of the Gram matrix and $\langle \lambda^{-1} \rangle$ denotes the average of the inverse eigenvalues over all frequencies (from $-N$ to N).

The result above, proven in App. B, will be leveraged throughout this study to gain insights into the geometrical quantities that matter for generalization of deep networks on symmetric datasets. Essentially, we will show that—on symmetric datasets—various kernels (RBF, MLP, CNN) lead to circulant (or approximately circulant) Gram matrices, allowing to use the spectral error formula (Eq. 1) to interpret the geometric factors on which generalization depends. Note that the spectral error can also be extended to multiple missing points (see App. B.1) but it leads to a less interpretable formula, so we do not consider this case.

The spectral error formula can also be generalized to *any finite group, including non-abelian groups*, see Theorem 15 in App. C. The resulting formula is necessarily more complex, requiring notions of representation theory and non-commutative harmonic analysis (generalized Fourier transform). For the sake of simplicity, we limit our treatment to the cyclic group in what follows.

4.3 A Simple Case Study: Gaussian Kernel Regression on a Circular Dataset

We first study the behavior of the Gaussian (i.e., RBF) kernel on a simple symmetric dataset in $\mathbb{R}^3$ (Fig. 2). We will see later that this case study captures all the relevant phenomenology for understanding the behavior of deep networks on datasets presenting symmetries.

Circular dataset in $\mathbb{R}^3$. Consider the following dataset of $2N$ points (see Fig. 2A for an illustration):

$$\mathcal{D} \equiv \{(g^i.x^A, +1)\}_{i=0}^{N-1} \cup \{(g^i.x^B, -1)\}_{i=0}^{N-1} \subset \mathbb{R}^n \times \mathbb{R},$$

where g is a representation of the generator of the cyclic group of order N, acting on $\mathbb{R}^3$ by usual matrix multiplication. This dataset is composed of two orbits of the same group, obtained for two different *seed* samples, x^A and x^B, which are labeled +1 and -1 respectively. We *order* the dataset so that points labeled +1 are interleaved with those labeled -1, while preserving individual orbit ordering.[3] For notational convenience, we denote $x_i^\bullet = g^i.x^\bullet$. The ordered dataset can be written:

$$\begin{aligned}\mathcal{D}_o &\equiv \{(g^0.x^A, +1), (g^0.x^B, -1), (g^1.x^A, +1), \cdots (g^{N-1}.x^B, -1)\} \\ &\equiv \{(x_0^A, +1), (x_0^B, -1), (x_1^A, +1), \cdots (x_{N-1}^B, -1)\}.\end{aligned}$$

We consider the following $\mathbb{R}^3$ representation of the generator g:

$$g \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix},$$

[3] While this ordering does not affect the results of kernel regression on a missing point (kernel methods are order-independent), it will allow us to diagonalize the Gram matrix via the Fourier transform, instead of a permutation of it, making the spectral formula directly interpretable in terms of the eigenvalues associated to each Fourier component.

where we define $\theta = 2\pi/N$. We consider the seed points

$$x^A \equiv \frac{\Delta}{2}\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad x^B \equiv -\frac{\Delta}{2}\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}0\\\cos(\theta/2)\\\sin(\theta/2)\end{bmatrix}, \tag{2}$$

over which we make g act by usual matrix multiplication. Successive applications of g onto the seed points generate the dataset. Note that the second seed point x^B is chosen such that the two orbits are *geometrically interleaved* (as depicted in fig 2A), which means that points of orbit A are equidistant from their two nearest neighbors in orbit B and vice-versa. This geometric interleaving is a necessary condition for the Gaussian kernel Gram matrix to be circulant (see below).

Gaussian (RBF) kernel regression on a circular dataset. We consider the following regression problem: remove any one point from the dataset $\mathcal{D}_o$, and estimate the value of y at the missing point. We solve this regression problem using *Gaussian process (GP)* kernel regression. We select the *Gaussian or Radial Basis Function (RBF)* kernel,

$$k_{RBF}(x_i, x_j) = \exp(-L^2\|x_i - x_j\|_2^2),$$

where L denotes the kernel's length scale.

Lemma 5 *The Gram matrix of a stationary kernel (a kernel that only depends on the Euclidean distance between pairs of input points) over a dataset made of two geometrically interleaved cyclic orbits (as defined above) is circulant.*

Proof First consider that the pairwise distance between points x_i and x_j in $\mathcal{D}_o$ is only function of their absolute index difference $|i - j|$. Since a *stationary kernel* only depends on the Euclidean distance between pairs of input points, it follows that a stationary kernel produces a *circulant* Gram matrix over $\mathcal{D}_o$. ■

The RBF kernel is a stationary kernel. The Gram matrix of the RBF kernel being circulant over $\mathcal{D}_o$, Proposition 4 applies, and prediction at the missing point is given by the spectral error (Eq. 1). This affords us a geometric interpretation of the generalization behavior of the kernel in the spectral (Fourier) domain.

Geometric interpretation The formula of the spectral error lends itself to a simple geometric interpretation (Fig. 2). Consider the spectrum obtained by applying the DFT to any one row of the kernel matrix.[4] We plot the inverse spectrum λ^{-1}. In this picture, the ratio in Eq. 1 corresponds to a ratio between areas, respectively those of a rectangle with height λ_N^{-1} for the numerator, and the area under the λ^{-1} curve for the denominator.

Let us now study the impact of the distance between orbits, Δ, on the prediction error (Fig. 3A-C). We vary Δ while keeping the number of points in one

[4] Such row is composed entirely of real entries. As a consequence, the spectrum is symmetric; we thus only show the positive half of the spectrum in the figure depictions, ranging from 0 to $N - 1$.

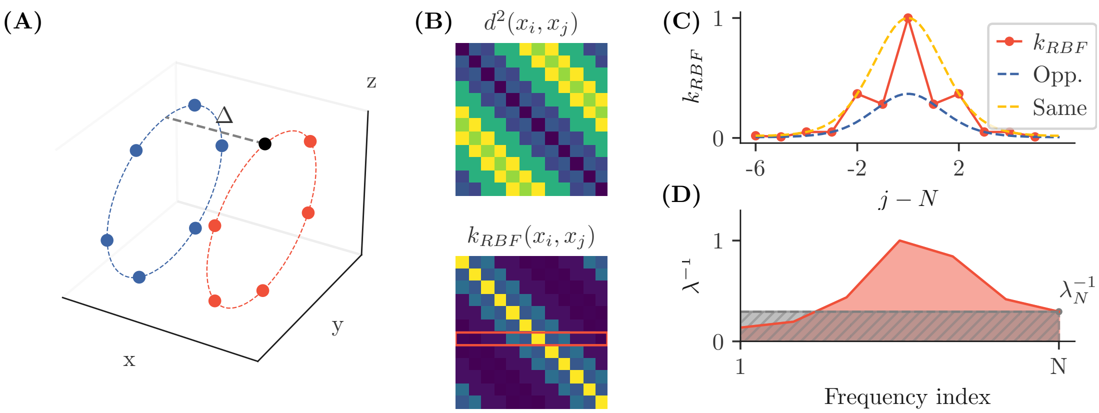


Figure 2: **Case study: Analysis of the generalization behavior of the Gaussian kernel on a circular-symmetric dataset.** **A**: A circular dataset is made of two sets of interleaved points in $\mathbb{R}^3$ belonging to two classes (denoted in red and blue). One point (in black) is left out during training, and we regress on it. **B**: The pairwise distance matrix between the points is circulant (above), leading to a circulant kernel matrix (below) obtained by applying the Gaussian (RBF) kernel function to it elementwise. We select the N-th row from it, highlighted by the red rectangle; due to the circularity of the matrix, this comes without loss of information. **C**: A plot of the selected row of the kernel matrix. The values of the kernel alternate between two limiting curves, representing respectively the "same label" and the "opposite label" kernel values. This alternation, due to the separation between classes, is akin to the presence of a high frequency component in the kernel function itself, were it computed over the classes perfectly interleaved ($\Delta = 0$). **D**: The prediction error of a Gaussian kernel on a missing point of a circular-symmetric dataset is a simple function of its spectrum (Eq. 1). The reciprocal (inverse) of the positive half of the DFT of the selected row of the kernel matrix is shown. The grey area corresponds to the numerator of the spectral error, while the red area corresponds to the denominator. The ratio of the two corresponds to the prediction error incurred by kernel regression using the chosen kernel. The presence of the aforementioned high frequency component in the kernel is manifested in the low value of λ_N^{-1}, leading to small error (i.e., increased class separability).

orbit, N, and the kernel's length scale, L, fixed. We compute the prediction error by solving kernel regression for varying Δ, and compare it with our formula for the spectral error (Fig. 3B). As expected by their mathematical equivalence, the two agree across the investigated range of values, and decrease as a function of Δ. This decrease aligns with the following intuition: pulling the classes apart makes prediction over the missing point by a local kernel easier.

In Fig. 3C, we show the geometric interpretation of the spectral error. Increasing Δ, by increasing the power of the highest frequency of the kernel matrix, skews the ratio in favor of

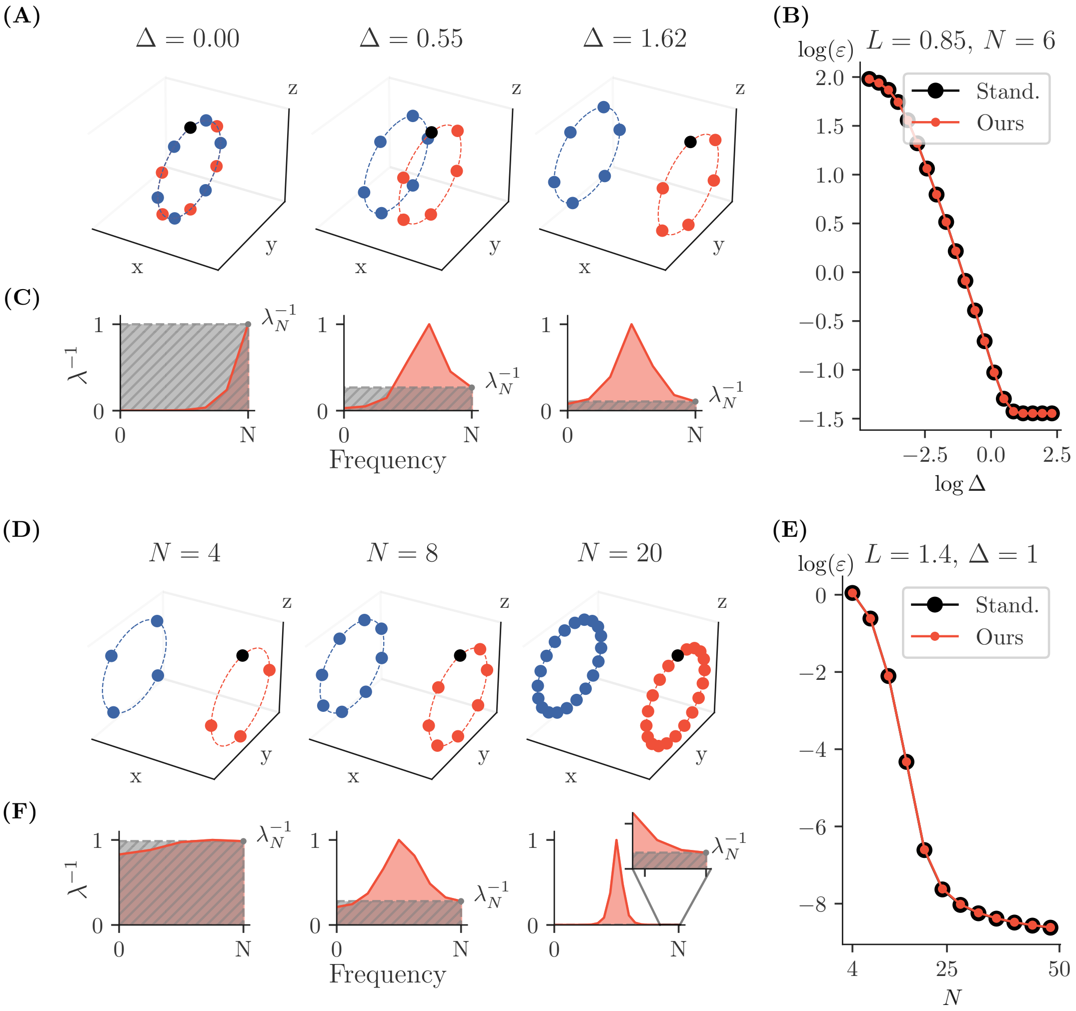


Figure 3: **Further geometric interpretations of the Gaussian kernel generalization behavior on a circular-symmetric dataset. A**: We progressively increase the separation Δ between two circular classes of points (respectively in blue and red). The leave-out point used for testing is in black. **B**: The prediction error (in black) of the RBF kernel on the leave-out point decreases as a function of Δ, as predicted by our spectral formula (in red). given in Eq. 1. **C**: Inverse spectra of the kernel matrix for different Δ. The grey (respectively, red) area corresponds to the numerator (resp., denominator) of Eq. 1. As Δ increases, the ratio between grey and red areas (i.e., the spectral error) progressively decreases, as a consequence of the last frequencies getting larger. **D, E, F**: Same as *A*,*B*,*C* where instead of increasing Δ we increase the number of points forming each class. The ratio between grey and red areas progressively decreases with number of points, as a consequence of the middle inverse frequencies progressively diverging.

the denominator in Eq. 1 (denoted by the red area). For $\Delta = 0$, the points are interleaved in the yz plane, and a kernel that would successfully generalize on the missing point would need to be dominated by its highest frequency (oscillating) component. This is not the case for the RBF kernel, whose spectrum is instead dominated by the low frequency components due to its local nature. As Δ increases, however, the kernel's spectrum effectively gains power in the highest frequency component, thus "aligning" its properties to the requirements imposed by the regression problem.

In summary, an increase in the distance between orbits Δ leads to an increase in the power of the highest frequency term λ_N of the kernel matrix, which leads in turn to better kernel generalization on the missing point.

We now turn to study the impact of the number of angles in an orbit N on the prediction error (Fig. 3D-F). We compute the prediction error by solving kernel regression for varying N, and compare it with our formula for the spectral error (Fig. 3E). As expected by their mathematical equivalence, the two agree across the investigated range of values, and decrease as a function of N. This decrease aligns with the following intuition: increasing the point density of an orbit makes prediction over a missing point easier.

In Fig. 3F, we show the geometric interpretation of the spectral error. Increasing N skews the ratio in favor of the denominator of Eq. 1 (denoted by the red area). The DFT projects an n dimensional signal over n discrete frequencies. For a local kernel, such as the RBF, which has most of its power slotted onto its first few low frequencies, sampling more points and then taking the DFT effectively means expanding the number of frequencies with low power, leading to the corresponding spectrum elements becoming vanishingly small. This in turn causes the average of the inverse spectrum to diverge, making the denominator of Eq. 1 grow.[5]

In summary, we see that increasing the number of angles along an orbit N leads to an increase in the average inverse spectrum $\langle \lambda^{-1} \rangle$ (denominator of Eq. 1), and thus to a decrease in generalization error on the missing point.

These geometric insights will prove useful in interpreting the generalization capabilities of deep networks on high dimensional datasets possessing symmetries.

4.4 Multi-layer Perceptrons on Rotated-MNIST

Building on the insights from the previous section, we now turn to a more realistic setting. We study the predictions of Multi-layer Perceptrons (MLP) in the Neural Tangent Kernel limit (NTK), on partial views of rotated-MNIST, a high-dimensional dataset containing a rotational symmetry.

Datasets and architectures We consider the MNIST dataset, and we augment it by means of (discrete) rotations. More precisely, for each MNIST digit, N images are generated by rotation in increments of $2\pi/N$, capturing the full rotational range. This way, we have access to a complete *rotational orbit* for each sample. This rotation transformation corresponds, up to pixel discretization effects, to the action of a particular representation g

[5] We note, however, that the presence of a nonzero Δ implies that the *highest* frequency terms in general, and λ_N in particular, remain large, leading to the "explosion in the middle" of the inverse spectrum, which keeps the numerator in Eq. 1 small.

of the cyclic group of order N (specifically, the regular representation), acting on the vector space $\mathbb{R}^{28\times 28}$ in which the greyscale MNIST images live. Additionally, we normalize the digits to be on the sphere, making their orbits comparable with each other.

We then create datasets by forming *pairs* of these orbits, each pair consisting of digits from different classes. Given these datasets, we define a regression problem by leaving out one of the points in the two orbits, and asking to find a predictor for this missing sample. Note that unlike in the more realistic scenario presented in Fig. 1, our classes here are constructed from a single seed sample and their orbit, and we only consider two classes. However, we will show in Section 4.5 that our conclusions on this simplified setup also apply to the multi-seed-per-class and multi-class problems, under further approximations.

We now focus on neural networks as our predictors. Specifically, we consider respectively a 1-hidden-layer and a 5-hidden-layer MLPs with ReLU activations (see App. D for more details on the architectures) in the NTK limit. In this limit, the predictions of the networks can be written as the result of kernel regression, where the kernel function is determined by the architecture of the network. Therefore, we can replicate the analysis performed in the previous section, where we replace the simple Gaussian kernel with a neural kernel. In the following, we refer to our kernel function as k_{NTK}, and the respective kernel matrix as K^{NTK}. In practice, we use the `neural-tangents` library (Novak et al., 2020) to compute the neural kernels.

Importantly, while this setup is partly analogous to the case study of the previous section (RBF kernel over a circular dataset), there are some key assumptions and approximations that need to be justified before we can apply the theory to this case.

Firstly, unlike the RBF kernel, the MLP kernel is not stationary, so Lemma 5 does not apply to the MLP kernel. However, we can show that the MLP kernel, in virtue of being a *dot product* kernel (Neal, 1996), also produces a circulant Gram matrix over an orbit of the cyclic group:

Lemma 6 *The Gram matrix of a dot product kernel is circulant over a single orbit of the cyclic group.*

Proof The cyclic group acts on a seed point linearly via its (matrix) representation R; due to the properties of the cyclic group, such representation is orthogonal. The dot product between data points thus only depends on their index difference, as indexed by the group action:

$$(x_i^A)^T x_j^A = (R^i x_0^A)^T (R^j x_0^A) = (x_0^A)^T R^{-i} R^j x_0^A = (x_0^A)^T R^{j-i} x_0^A$$

As the kernel of a dot product only depends on the scalar product of its inputs, it thus also only depends on the index distance between the data points:

$$K_{ij} = k(x_i^A, x_j^A) = k((x_i^A)^T x_j^A) = k((x_0^A)^T R^{j-i} x_0^A),$$

and thus the (i, j) entry of the Gram matrix depends only on the difference between the two indices $(i - j)$, modulo N because of the properties of the rotation matrix. The Gram computed over a single orbit is thus circulant. ■

Next, we remind the reader that the datasets we consider here are composed of two orbits, obtained by applying the cyclic group action to two different MNIST digits. Unlike in the synthetic dataset of the previous section, these two orbits are not geometrically interleaved, as we have no control over the position of the seed points given by the MNIST digits. This leads us to make the following approximation and adjustment.

Approximation: Circularity of the kernel matrix over the dataset made of two orbits MNIST digits from different classes are in general not *geometrically* interleaved, i.e., a point in one orbit is not equidistant from the "neighboring" points in the other orbit. This can be written as the following dot product condition:

$$x_i^A \cdot x_i^B \neq x_i^A \cdot x_{i+1}^B.$$

Furthermore, the angular distance between first neighbors in each orbit is not necessarily the same, i.e.,

$$x_i^A \cdot x_{i+1}^A \neq x_i^B \cdot x_{i+1}^B.$$

Indeed, even though the seed images are all normalized to be on the sphere, the angular distance between first neighbors on an orbit still depends on how close the seed image is to the stabilizer of the group action. Take for instance a seed image with perfect central symmetry (e.g., a perfect 'o'); all rotations of the seed image are the seed image themselves, and thus the angular distance is 0. Conversely, for a "Dirac delta"-like image (i.e., an image that is nonzero at a single point, assuming infinite pixel resolution), all rotated images are orthogonal to each other, setting their angular difference to 1. The angular distance between first neighbors on a same orbit thus sits between 0 and 1, depending on the smoothness of the image with respect to the group transformation.

As a consequence, the following NTK matrix is in general not circulant:[6]

$$K^{\mathrm{NTK}} = k_{\mathrm{NTK}} \left(\begin{bmatrix} x_0^A \cdot x_0^A & x_0^A \cdot x_0^B & x_0^A \cdot x_1^A & x_0^A \cdot x_1^B & x_0^A \cdot x_2^A & \cdots \\ x_0^B \cdot x_0^A & x_0^B \cdot x_0^B & x_0^B \cdot x_1^A & x_0^B \cdot x_1^B & x_0^B \cdot x_2^A & \cdots \\ x_1^A \cdot x_0^A & x_1^A \cdot x_0^B & x_1^A \cdot x_1^A & x_1^A \cdot x_1^B & x_1^A \cdot x_2^A & \cdots \\ x_1^B \cdot x_0^A & x_1^B \cdot x_0^B & x_1^B \cdot x_1^A & x_1^B \cdot x_1^B & x_1^B \cdot x_2^A & \cdots \\ x_2^A \cdot x_0^A & x_2^A \cdot x_0^B & x_2^A \cdot x_1^A & x_2^A \cdot x_1^B & x_2^A \cdot x_2^A & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \right)$$

To retrieve the setup outlined in the previous section, we resort to an *ex-post* circularization procedure. We consider the matrix $\tilde{K}^{NTK}$ that is obtained by taking the diagonal-wise average of the kernel matrix K^{NTK}:

$$\tilde{K}_{ij}^{NTK} = \frac{1}{2N} \sum_k K_{(i+k)\%(2N),(j+k)\%(2N)}^{NTK}.$$

[6] We can note that the first diagonal (denoted in black) is constant in virtue of the fact that we normalize each data point to be on the sphere.

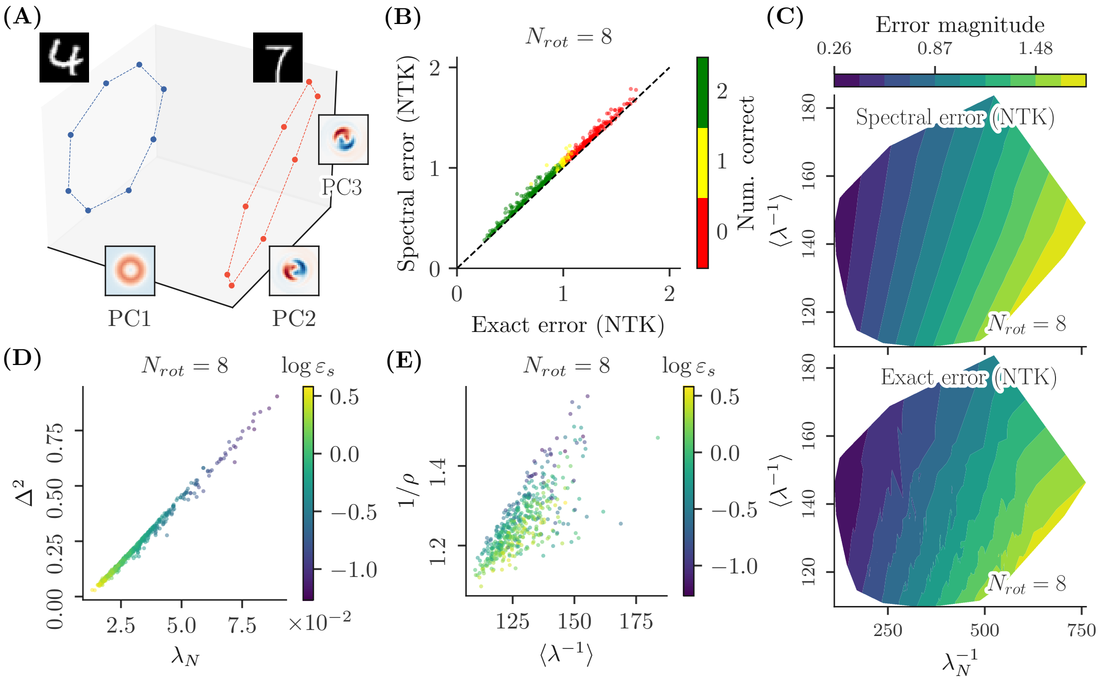


Figure 4: **Analysis of the prediction behavior of a MLP on pairs of orbits from rotated-MNIST. A**: We take samples of two different MNIST digits and generate their rotation orbits. The task is to predict the label value of a leave-out point in one of the two orbits. We use angle steps of 45 degrees, amounting to $N_{rot} = 8$. We show the points in a reduced-dimensional space obtained by performing PCA on the dataset. **B**: Scatter plot of the NTK error computed in the standard way (exact) against our spectral error (Eq. 1). Each dot corresponds to a different dataset, obtained by randomly picking pairs of digits of different classes. The color coding corresponds to the number of classification errors incurred by the symmetrized NTK regression, obtained by excluding a point from class A, then from class B, and counting the number of classification errors (0, 1 or 2), understood as a disagreement in sign between NTK prediction and label of the missing point. **C**: Comparison between the spectral and exact NTK error across different values of λ_N^{-1} and $\langle\lambda^{-1}\rangle$. **D**: Comparison between the values of λ_N and Δ^2. The former is the highest frequency component of the neural kernel matrix (inverse of the numerator in Eq. 1), while the latter is the distance in pixel space between the averages (centroids) of the two orbits. **E**: Comparison between the values of $\langle\lambda^{-1}\rangle$ and $1/\rho$ (see main text).

Such matrix is by definition circulant. The practical meaning of this procedure is twofold: the averaging of *even* diagonals imposes that both classes' feature embeddings are "equally spaced" along the orbit, while the averaging of *odd* diagonals (together with the intrinsic symmetry of the kernel matrix) imposes that the feature embeddings are interleaved in the

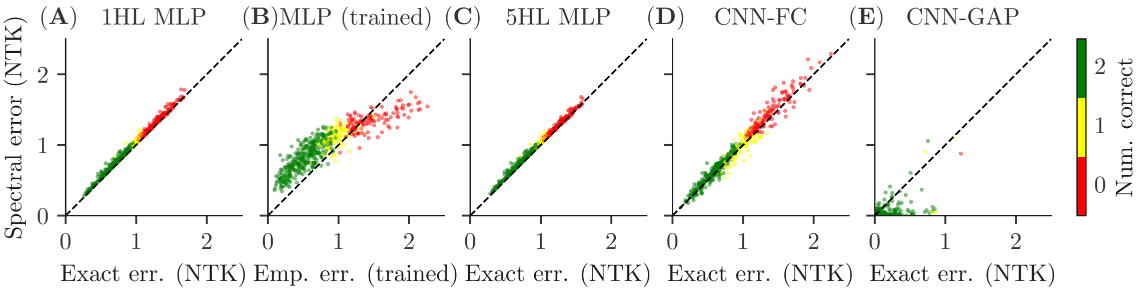


Figure 5: **Spectral error matches exact NTK error across various architectures and for finite-width networks, on rotated-MNIST orbit pairs**: **(A)** a MLP with 1 hidden layer, **(B)** a finite-width 1-hidden layer MLP trained with Adam, **(C)** a MLP with 5 hidden layers, **(D)** a ConvNet with a fully-connected last layer, **(E)** a ConvNet with global average pooling at the last layer. In this case, the assumptions of the theory are too crude to capture empirical phenomenology (see text). Color coding as in Fig. 4B.

sense that $\tilde{K}^{NTK}_{i,i+1} = \tilde{K}^{NTK}_{i,i-1}$. We stress that this circularization procedure is not justified a priori. However, we empirically show that it is a realistic approximation for analyzing the interplay between symmetric datasets and deep networks.

Adjustment: Symmetrization of the NTK error w.r.t. to class Because of the asymmetry between orbits, removal of a point from one orbit is not in general equivalent to removal of a point from the other orbit. Yet, our formula for the spectral error, operating on the circularized kernel matrix, is intrinsically symmetric. To restore this interchangeability in the standard NTK error, we consider in the following a *symmetrized NTK error*, which we obtain by averaging the NTK prediction errors that arise by removing a point from either one orbit or the other. It is this symmetrized NTK error that we compare with our spectral error.

Results Equipped now with a circulant kernel matrix, we can once again refer to the spectral error formula of Eq. 1 for the estimation of the prediction error associated with performing kernel regression under the kernel matrix $\tilde{K}^{NTK}$.

We find that the spectral error tracks the symmetrized NTK error across all sampled pairs (Fig. 4A-B). This agreement holds across the whole range of explored values for both λ_N and $\langle \lambda^{-1} \rangle$ (Fig. 4C), and regardless of the number of rotation angles (App. F). Moreover, this agreement qualitatively holds for finite-width, trained networks (Fig. 5B and App. G), as well as for deeper (5-layer) MLPs (Fig. 5C and App. G). This suggests that the circularization procedure preserves the overall structure of the problem, and allows us to study generalization through a geometric lens as in the previous section.

Geometric interpretation The quantities involved in the spectral error (Eq. 1) are computed in kernel space, not in input (i.e., dataset) space, which makes their interpretation difficult. However, for simple kernels such as the MLP kernel, we will now show empirically that these quantities correlate with geometric quantities in input space, allowing us to understand the geometric factors underlying generalization.

Firstly, the highest frequency eigenvalue λ_N correlates with the orbit separation in input space Δ (Eq. 2). Indeed, the values of λ_N and the euclidean distance between orbit averages in input space are, for the explored ranges, in an approximate linear relationship (Fig. 4D). *As a consequence, one can see how the separation between the orbits is a key indicator of prediction error for the MLP kernel.*

Secondly, we can map the average reciprocal spectrum $\langle \lambda^{-1} \rangle$ to a measure of orbit density in input space, computed as the *inverse of the orbit radius.* The orbit radius is computed as the distance of any orbit sample to the orbit centroid, in the high-dimensional dataset space.[7] We see that these two quantities are in an approximately linear relation (Fig. 4E).

These relations enable a geometric interpretation of the quantities that determine generalization success: generalization success depends on the distance between classes Δ and the density of orbits. Note, however, that the observed mapping between spectral quantities of the kernel and geometric quantities in input space is empirical and not guaranteed to hold for all architectures. In particular, we show later that this equivalence breaks for a convolutional neural network with global average pooling at the last layer, making the interpretation of the spectral formula terms less direct.

4.5 Extension to Multiple Seeds per Class and Multiple Classes

Until now, we have only considered a simplified scenario in which the datasets are composed of a single orbit per class, and where there are only two classes. We now ask whether the theory can also be adapted to more a realistic scenario, where datasets are composed of multiple seed images per class, and where there are multiple classes, only one of which has a missing angle during training. This setting is the one we presented at the start of this study (Fig 1). Below, we show that our framework can be extended to this scenario.

Extension to multiple seed points per class First, we describe how our framework can be extended to multiple seeds per class and two classes. Since our spectral formula only applies to datasets made of two orbits (one for each class), we make the simplifying assumption that pairs of orbits interact *linearly* to predict the average error on a missing point. In other words, we average the spectral error computed over each possible pairs of orbits taken across classes.

Specifically, for all pairs of seeds that can be formed across the two classes, we compute a spectral error according to Eq. 1. By doing so, we are considering the case of a missing point in either of these two orbits. We then average the error over all the pairs, obtaining a single, average spectral error.

We compare this average spectral error against the exact NTK error, computed over the entire dataset. We obtain the exact error by running NTK regression on the dataset that contains all orbits of both classes, excluding one angle from all orbits of one class. On said angles, we compute the NTK regression error, and average it over missing points from all

[7] The orbit radius is a measure of how close the seed image is from the stabilizer of the group action. For example, a seed image that is rotation invariant, such as a perfect ‘o’, is situated on the stabilizer of the rotation group action, leading to a null radius.

orbits of that class. We then compute its symmetrized version by swapping the role of the two classes, and averaging the two error values.

We test the agreement between the pair-averaged spectral error and symmetrized NTK error across many trials, where each trial corresponds to a different dataset, obtained by taking a number of seeds ($N_{\text{seed}} = 13$) from each of two different, randomly picked MNIST digit classes. We limit the datasets to contain 13 seeds per class for computational reasons. We find empirically that the average spectral error correctly predicts exact NTK error across a large number of such trials (Fig. 6A, extended version in App. H). We note that the spectral error does not capture well the magnitude of the exact error anymore (in reason of the additional assumptions needed in the multi-seed case). However, the spectral error still correlates well with the exact error across datasets.

In conclusion, our spectral theory, which simply considers the average pairwise interaction between orbits of different classes, predicts well the generalization behavior of infinite-width networks on a dataset comprising multiple seeds per class.

Extension to multiple classes We now describe how our framework can be extended to the 10 classes of MNIST. We stress that, in principle, NTK regression is not suited to predict the results of a network that is trained with a cross-entropy loss on multiple classes. However, we show that we can adapt our spectral theory to this scenario, and qualitatively model the results of such training. We do so by employing a *one-versus-many* strategy.

Consider an orbit from class A. Form all pairs with orbits of another class B (there are $N_{\text{seed}} = 13$ such pairs). Average the spectral prediction obtained from all these pairwise comparisons. Repeat this comparison with all other classes C, D, E, etc. We thus obtain a prediction for a missing point in the orbit of class A against every other *class*. If *all* of these class-wise predictions are correct, we consider the network prediction on this orbit of class A to be correct. We extend this procedure to all orbits of class A, and count the percentage of correctly classified orbits for that class. We repeat this procedure for every possible leave-out class, and report the average accuracy of our classifier over all leave-out classes.

We compare the predicted accuracy resulting from this procedure, to the empirical accuracy of a finite-width 2-hidden-layer MLP trained with Adam and a cross-entropy loss (see details of architecture in App. D.3), on a version of rotated-MNIST comprising $N_{\text{seed}} = 13$ seeds per class (we limit the number of seeds per class to 13 because of the prohibitive computational cost of computing NTK regression on larger datasets).

The spectral accuracy qualitatively matches the empirical accuracy of the trained network as we vary the number of points composing the orbits (Fig. 6B, extended version in App. Fig. 14). The spectral accuracy curve also mirrors the empirical curves obtained by training various architectures (MLP, CNN, ViT) on a full version of rotated-MNIST at the beginning of this study (Fig. 1). We note that classification accuracy increases with the number of sampled angles until it saturates to perfect classification accuracy. The saturation effect comes from the fact that we evaluate the network on a classification task. When the elements of the orbits are dense enough in kernel space, the error in prediction becomes small enough that the test point is systematically correctly classified.

In conclusion, our spectral theory, which only considers pairwise interactions between seeds across classes, and lends itself to a simple geometric interpretation of the factors

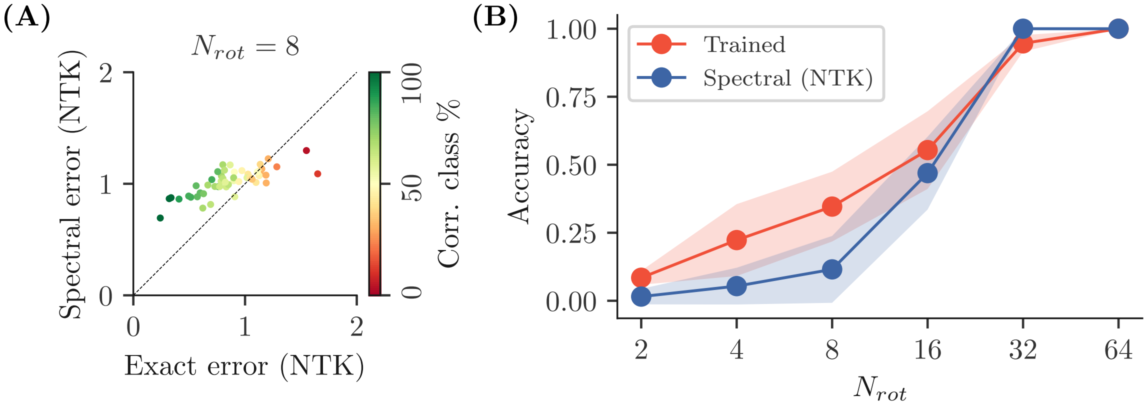


Figure 6: **Application of the spectral theory to a MLP trained on a subsection of rotated-MNIST comprising multiple seeds per class and multiple classes. A**: For datasets comprised of two classes of rotated-MNIST, comprising multiple seeds each ($N_{\text{seed}} = 13$), we compare the average spectral error, obtained by averaging over all pairings of orbits in the dataset, with the symmetrized NTK prediction error, computed over all possible missing points. Each dot in the scatter plot represents a different dataset, drawn by randomly selecting two classes from MNIST, and randomly selecting 13 seed images per class. The color coding reflects the percentage of seeds (of both classes) for which the NTK regression gives a correct prediction, understood as agreeing in its sign with the label of the missing points. **B**: On a multi-class (10 classes of MNIST), multi-seed-per-class ($N_{\text{seed}} = 13$) version of rotated-MNIST, we compare the generalization accuracy predicted by our multi-class adapted spectral error, with the one of a normally trained MLP (2 hidden layers, trained with a cross-entropy loss). As the number of points in the orbits increases, both trained and spectral accuracies increase on the classification task, progressively and similarly, suggesting that no mechanism for symmetry learning is present for finite-width trained networks that would be unaccounted for by the spectral theory.

leading to correct generalization (namely class distance and orbit density), qualitatively recapitulates the lack of generalization of conventionally trained finite-width networks on a realistic dataset, rotated-MNIST, comprising multiple classes and multiple seeds per class. Moreover, both theory and experiments show that the error is a progressive function of these two quantities, and that there isn't a phase transition where the network would suddenly learn a function that captures the symmetry of the problem. Taken together, these results provide strong evidence that the ability of conventional deep networks to generalize on symmetric datasets is essentially dictated by local geometrical properties of the datasets, and that no specific mechanism exists that would allow deep networks to learn from examples the non-local, symmetric structure of a prediction problem.

4.6 Extension to Equivariant Architectures

We here study how the interplay between dataset symmetries and equivariant architectures affects generalization.

For simplicity and concreteness, we focus our study on spatially convolutional neural networks. We distinguish two types of convolutional architectures: (1) convolutional architectures where the last layer is fully connected: these architectures do not ensure full invariance to translation; (2) convolutional architectures where the last layer performs global average pooling, ensuring invariance to translation.

We also distinguish two cases of data symmetries: (1) when the symmetry in the data corresponds to the one encoded in the equivariant architecture; (2) when it does not, and the architecture is equivariant to another, different symmetry. We consider a dataset with translational symmetry to illustrate the first case (a convolutional neural network is equivariant to translations) and a dataset with rotational symmetry to illustrate the second case (a convolutional neural network is not equivariant to image rotations).

Through these examples, we build a framework that should be easily extensible to characterize how equivariant architectures and dataset symmetries interact in general.

Below, we provide formal definitions for the aforementioned datasets and network architectures, and then proceed to state the results.

4.6.1 Definitions

Definition 7 (Dataset with translational symmetry) *A dataset with translational symmetry is composed of seed images and all their translations. For the purpose of the proofs to follow, we will focus on a single orbit of this dataset, which consists of a single seed point $x_s \in \mathbb{R}^{n\times n}$ and all of its translations:*

$$\mathcal{O}_T = \{g_T^0.x_s, g_T^1.x_s, \cdots g_T^{n-1}.x_s\},$$

where the translation operator g_T acts on images by circularly shifting them along one of the dimensions. For pixel coordinates (i_x, i_y), and the corresponding value of the pixel $x(i_x, i_y)$, we write:

$$g_T.x\left(\begin{bmatrix} i_x \\ i_y \end{bmatrix}\right) = x\left(\begin{bmatrix} (i_x+1) \bmod n \\ i_y \end{bmatrix}\right)$$

Definition 8 (Dataset with rotational symmetry) *A dataset with rotational symmetry is composed of seed images and all their rotations in C_4 (we limit ourselves to 4 cardinal rotations to avoid definitional problems of image rotation on discrete pixel grids). We will focus on a single orbit of this dataset, which consists in a single seed point $x_s \in \mathbb{R}^{n\times n}$ and all its rotations:*

$$\mathcal{O}_R = \{g_R^0.x_s, g_R^1.x_s, g_R^2.x_s, g_R^3.x_s\}$$

where the rotation operator g_R permutes pixel coordinates (i_x, i_y) as follows:

$$g_R.x\left(\begin{bmatrix} i_x \\ i_y \end{bmatrix}\right) = x\left(\begin{bmatrix} i_y \\ n - i_x \end{bmatrix}\right)$$

Definition 9 (Fully connected convolutional network (FC)) *A fully connected convolutional network is a network $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ parameterized by:*

$$f_{FC}(x) = A\frac{1}{\sqrt{k}}\phi(B \circledast x)_v,$$

where $A \in \mathbb{R}^{1\times n^2 k}$, $B \in \mathbb{R}^{k\times 1\times 3\times 3}$, $\circledast$ denotes the spatial convolution operation, and for any matrix $u \in \mathbb{R}^{n\times n}$, $u_v \in \mathbb{R}^{n^2}$ denotes the vectorization (i.e., flattening) of u. This network first applies a convolutional layer to the data, then flattens the resulting representation into a vector, and passes it through a fully connected layer. We assume circular padding and stride of 1. In experiments we use a filter of size 3×3.

Definition 10 (Global Average Pooling convolutional network (GAP)) *A GAP network is a convolutional network with global average pooling at the last layer. This network is invariant to discrete translations. The network $f_{GAP} : \mathbb{R}^{n\times n} \to \mathbb{R}$ is parameterized by:*

$$f_{GAP}(x) = \frac{1}{\sqrt{k}n^2}\sum_k \sum_{i_x} \sum_{i_y} A_{1k}\phi(B_{k,i_x,i_y} \cdot x)$$

where $A \in \mathbb{R}^{1\times k}$, and $B \in \mathbb{R}^{k\times 1\times 3\times 3}$ with $B_k \in \mathbb{R}^{1\times 1\times 3\times 3}$ indexing filter k of B. $B_{k,i_x,i_y} \in \mathbb{R}^{1\times 1\times n\times n}$ is obtained by centering B_k at coordinates (i_x, i_y) of an $n \times n$ grid with periodic boundary conditions, and filling the remaining entries with zeros. Then, the dot product is understood as the sum of the elementwise multiplications of all entries of x and B_{k,i_x,i_y}. We remark that this operation is effectively an alternative way of describing a convolution with filter bank B, but this indexing choice proves useful in the proofs. After applying a convolutional layer to the data, this network averages the resulting representation across each of the k output channels, and then takes a linear combination of these averages using a fully connected layer.

4.6.2 Theoretical Results on Equivariant Architectures

We showed previously that the generalization behavior of MLPs on a symmetric dataset, rotated-MNIST, is well captured by the spectral error, a quantity computed from the Fourier components of the MLP kernel over the (orbits of the) cyclic group of interest. Crucially, the derivation of this result relied on the circulant structure of the kernel matrix, which in turn was a consequence of the dot product nature of the MLP kernel.

Convolutional neural networks, however, are *not* dot product kernels, as pixel proximity plays a role in the kernel similarity between two images (Arora et al., 2019). Nevertheless, here we show that the kernel matrix of convolutional neural networks *is also circulant* over common representations of the cyclic group, regardless of whether the equivariance matches the symmetry of the data or not. We thus get that the same spectral theory of generalization applies to these equivariant architectures. Furthermore, a well-known special case arises when the network is designed to be fully invariant to the symmetry of interest, which is also well captured by our spectral theory.

First, we study the interplay between an equivariant architecture and its matching symmetry.

Proposition 11 *The kernel matrix of a fully connected convolutional network K_{FC} over a translation orbit O_T is circulant. Moreover, this kernel matrix is in general not constant or rank-deficient.*

See App. E for a proof of this proposition and the propositions below.

The kernel matrix of a fully connected convolutional network being circulant over a translation orbit, it can be analyzed in the Fourier domain, as done in the previous sections for a MLP. The same spectral formula for generalization error can thus be derived for this architecture and symmetry. Moreover, the kernel matrix is in general not rank-deficient on the translation orbit. This implies that, *a priori*, no inverse eigenvalue will diverge in the denominator of Eq. 1. Were this to happen, the spectral error would go to 0, i.e., the network would achieve perfect generalization. The same conclusion about the inability of a MLP to extrapolate symmetries to partially observed classes thus holds for a fully connected convolutional network on a dataset with translational symmetry.

Proposition 12 *The kernel matrix of a global average pooling convolutional network K_{GAP} over a translation orbit O_T is constant.*

The kernel matrix of a global average pooling convolutional network is not only circulant, but also constant. The kernel matrix computed over a dataset made of two orbits of two different classes is thus blockwise-constant in 2x2 blocks, one for each orbit, and by consequence rank-deficient. This rank-deficiency ensures that some of the eigenvalues of the kernel matrix are null, leading the denominator of the spectral error (Eq. 1) to diverge and thus perfect generalization (0 error).[8] We thus recover the well-known fact that a convolutional network with global average pooling at the last layer makes predictions that are by construction invariant to translations, and thus correctly generalizes from a partial view of a class orbit.

Second, we study the interplay between an equivariant architecture and a mismatching symmetry.

[8] This is true in the general case where the last frequency (at the numerator of the spectral formula) has non-zero power, i.e., the two classes are not fully collapsed in kernel space.

Proposition 13 *The kernel matrix of a fully connected convolutional network K_{FC} over a rotation orbit O_R is circulant, but in general not constant or rank-deficient.*

For a fully connected convolutional network applied to a dataset with rotational symmetry, the kernel matrix over an orbit remains circulant. Therefore, the same spectral theory of generalization applies as before. The kernel matrix is in general neither constant nor rank-deficient, and thus generalization on the missing point is not guaranteed to succeed.[9]

Proposition 14 *The kernel matrix of a global average pooling convolutional network K_{GAP} over a rotation orbit O_R is circulant, but in general not rank-deficient or constant.*

The implication of this last proposition is that (1) our spectral theory applies to this scenario as well, and (2) unlike for translations, the global average pooling layer does not guarantee perfect generalization on a rotation orbit.

4.6.3 Empirical Results on Equivariant Architectures

Empirically, we repeat the analyses previously performed on MLPs using convolutional architectures and obtain essentially the same results.

Fully connected (FC) convolutional network on rotated-MNIST We first analyze the behavior of a fully connected (FC) convolutional network, on the task of classifying a missing point from a dataset composed of two orbits, each generated from 8 rotations of a seed MNIST digit. The FC kernel matrix over a rotation orbit is circulant, as predicted by Prop. 3, allowing us to apply the spectral formula as we did for a MLP. The spectral error predicts well the symmetrized NTK error, across all pairs of orbits tested (Fig. 5D). As with a MLP, the numerator and denominator values of the spectral error are approximately proportional to the distance between classes in input space and their average orbit density respectively (Fig. 7), allowing the same interpretation of the factors that lead to successful generalization as for the MLP.

Global average pooling (GAP) convolutional network on rotated-MNIST We next consider a global average pooling convolutional network, on the same task and datasets. Its kernel matrix over a rotation orbit is also circulant, as predicted by Prop. 4, allowing us to apply the spectral theory. The prediction errors are in general remarkably low (but, crucially, not 0) for this architecture on these datasets (Fig. 5E).

While we lack a comprehensive theory for this success, we speculate that the GAP convolutional network finds similarities in the local structure of images belonging to the same rotation orbit, allowing it to outperform other architectures. Interestingly, we find that a GAP convolutional network architecture where the filters don't overlap (i.e., the stride matches filter size) fails to correctly classify some pairs of orbits (App. Fig. 16).

[9] This result, however, does not preclude a neural *classifier* from classifying the point correctly. Indeed, if the error is less than 1, a classifier would classify the point correctly (since the prediction is closer to the correct label than the other label which is -1 in our setup). Whether the neural classifier will classify the point correctly depends on the specific interplay between orbit density and class separation in kernel space, which depends on geometric properties of the data and of the architecture, as described by the spectral formula.

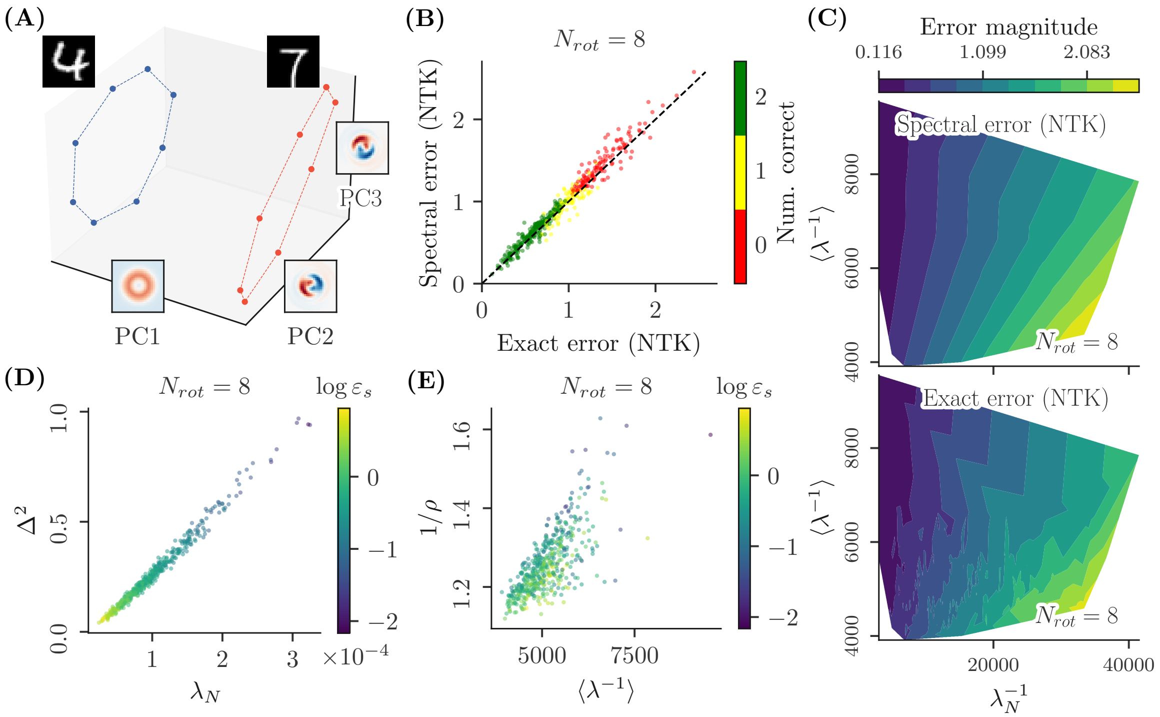


Figure 7: **Analysis of the prediction behavior of a fully connected convolutional network on pairs of orbits from rotated-MNIST.** Same caption as Fig. 4.

Moreover, the agreement between exact NTK error and spectral error that can be observed for other architectures is here worse, although still reasonable (both methods predict low error). Our interpretation of why spectral error is not a very good approximation of exact NTK error here, is that the post-hoc circularization of the kernel matrix may reshape its structure too drastically. For instance, the geometric interleaving between the two orbits that is assumed by the circularization, may not be a good approximation in this kernel space.

Finally, as shown in App. I.1, the numerator and denominator values of the spectral error do not map well to the distance between classes in input space and their average orbit density. *This illustrates the important fact that, for some neural architectures, the spectral quantities computed in kernel space are not direct correlates of simple geometric quantities in input space, and cannot be interpreted as simply. Nevertheless, even in these cases, the error on a missing point is given by the spectral formula (with the approximations mentioned), where the quantities in the formula represent geometric quantities in kernel space.*

Convolutional networks on translated-MNIST We next analyze the behavior of convolutional networks on pairs of orbits of translated-MNIST, a version of MNIST where the digits are translated along the image x-axis, with periodic boundary condition (App. I.2). Using a fully connected convolutional network, we do not see good generalization, as predicted by Prop. 1, and in good quantitative agreement with the spectral formula. Using a global average pooling convolutional network, we see perfect generalization of the network on the leave-out class (error is 0 to numerical precision), as expected from the fact that this architecture is invariant to translation. In our spectral formula, this perfect generalization is realized by the fact that the denominator of Eq. 1 diverges, as a consequence of the kernel matrix being rank-deficient (a consequence of Prop. 2).

In conclusion, we find that the generalization behavior of convolutional architectures on symmetric datasets is governed by the same formula and depends on the same geometric quantities as in the MLP case (namely orbit separation and density in kernel space). For highly non-linear architectures such as networks with a global-average pooling at the last layer, it is important to note that the geometric quantities in kernel space entering our spectral error are no longer trivially correlated to the same geometric quantities in input space, making the interpretation of these quantities kernel-specific. Finally, we find that only when the network is designed to be fully invariant to the symmetry of interest (e.g., GAP-CNN for translations), can it be theoretically guaranteed that the network will systematically generalize the symmetry correctly to unseen classes.

5. Discussion

In this work we establish a theory on when—and to what extent—deep networks are capable of learning symmetries from data. We find that the generalization behavior of conventional networks trained with supervision on datasets presenting a cyclic-group symmetry is captured by a simple ratio of inverse kernel frequency powers. Our analysis of this formula sheds light on the limitations of conventional architectures trained with supervision to learn symmetries from data. In particular, we find that they are generally unable to extrapolate symmetries observed exhaustively on some classes to other partially sampled classes. Accurate generalization is only possible when the local structure of the data in kernel space (determined by network architecture) allows for correct generalization. In other words, conventional networks have no mechanism to learn symmetries that have not been embedded in their architecture *a priori* through equivariance.

What are the practical implications of our results? There may be different remedies to the issues described above, which may also depend on the specific constraints of the problem. We outline here these solutions briefly before entering a more thorough discussion. First, resorting to very large datasets or having prior knowledge about the symmetries of the problem so as to build them into the model via data augmentation or equivariance may suffice in some cases. When this is not possible, introducing soft biases encouraging the network to find equivariant structures may be a fruitful alternative. Pretraining with self-supervised learning losses using the symmetries of interest as augmentation may also alleviate the problem of generalization, but it would require further studies to understand how such pretraining affects learned representations. We cannot exclude that in the feature-rich learning regime, symmetries can be learned and generalized—at least in some cases. Finally, there may be some entirely new learning procedures which could somehow respect the symmetries of the data without explicitly building these symmetries into the model.

Could neural networks learn symmetries outside the NTK regime? Our theory is only valid in the NTK regime, also sometimes referred to as the 'lazy regime', which assumes (1) infinite width, and (2) a certain scaling of the parameters with width (weights scale as $1/\sqrt{N}$ where N is layer width). We found in our experiments that the theory also recapitulates well the training dynamics of finite-width networks using Pytorch default scaling of weights (Kaiming Uniform, weights scale as $1/\sqrt{N}$). However, we cannot exclude that for other initialization schemes, datasets (e.g., CIFAR-10, ImageNet) or network architectures (e.g., deeper MLPs), finite-width networks could successfully learn the symmetry of the problem. Furthermore, other regimes of weight scaling have been studied in the infinite limit, for example the 'mean-field' regime (where weights scale as $1/N$) (Bach, 2017; Mei et al., 2018; Chizat and Bach, 2018), the 'high-dimensional' regime (Saglietti et al., 2022), and the 'dynamical mean-field' regime (Bordelon and Pehlevan, 2022; Yang et al., 2024a). These regimes, often referred to as 'feature learning' or 'active' regimes, are characterized by a kernel that evolves in time, and thus learns features. This leads to different generalization behaviors both in the finite-width (Geiger et al., 2020) and infinite-width limit (Bordelon and Pehlevan, 2022). It would be interesting to study symmetry learning in these regimes. In a recent study, Jacot et al. (2025) showed that deep networks in the active regime are able to learn data symmetries. However, their definition of symmetry is that of an invariant subspace warped by a non-linear function (through *low-index functions*, i.e., functions of the form $f(x) = g(Ax)$ where A is a low rank matrix). In contrast, we define a symmetry as the *non-local, linear* action of a group on a dataset.

Our results may seem at odds with the notion that deep networks are biased to find simple solutions (Shah et al., 2020; Ortiz-Jimenez et al., 2020; Zhang et al., 2021; Power et al., 2022; Humayun et al., 2024). In our study, networks are blind to the simple symmetric structure of the data. Were they to notice this simple structure, they could achieve data-efficient generalization. Importantly, results showing that networks have a bias towards simplicity typically define simplicity as a notion of local structure in the data. These studies show that networks find the smoothest possible function capturing the training set. Here, however, the symmetric structure of the data is not a local structure: rotated versions of a digit are not close in pixel space to the upright digit, and they may even be closer to a different digit class altogether. For example, a '4' upside down might look more like a '6' than like another upright '4'. It is thus expected that networks with a bias for smooth solutions will fail on such task, in the absence of a mechanism to detect symmetries. Networks biased to find the *shortest description* of a dataset (in the sense of Kolmogorov complexity) may perform better on this task (Valle-Perez et al., 2019), but progress in this area is limited. Bayesian model selection approaches may be a promising avenue to find such shortest description (van der Wilk et al., 2018; Immer et al., 2022; van der Ouderaa and van der Wilk, 2022; van der Ouderaa et al., 2023), and have recently been applied to learning data symmetries (van der Ouderaa et al., 2024).

On the other hand, our findings are compatible with the scaling laws observed when training deep networks (Kaplan et al., 2020; Bahri et al., 2024). Scaling laws show that deep networks continuously improve on generalization as the number of samples in the training set increases. Mapped to our setup, this corresponds to our observation that network performance gradually increases with the number of angles sampled from the group orbits. We see no abrupt change in generalization performance that would indicate that the network captures or "groks" the symmetry invariance, even with very long training times. **The absence of a mechanism to capture data symmetries could play a role in the notable data-inefficiency of current deep learning approaches**, inefficiency demonstrated by the evergrowing datasets used for training them (e.g., in computer vision JFT-300M (Sun et al., 2017), IG-3.6B (Singh et al., 2022) and LAION-5B (Schuhmann et al., 2022)). As the number of symmetric transformations present in the data increases (e.g., image rotation, translation, scaling, etc.), we only expect the generalization difficulty to increase, as this leads to a combinatorial explosion of possible transformations (Schott et al., 2022). This combinatorial explosion could partly be responsible for the brittleness of current deep learning approaches on edge cases, as observed for instance in Abbas and Deny (2023), where it is shown that deep networks for vision fare particularly poorly on *combinations* of symmetric transformations such as scaling and rotation.

Many methods have been proposed to learn symmetries from data. For example, some propose to leverage more supervision by pairing images undergoing a symmetric transformation during training. This is the case, for instance, of self-supervised learning approaches via joint embedding (Balestriero et al., 2023), and of autoencoder approaches (Dupont et al., 2020; Connor and Rozell, 2020; Keller and Welling, 2021a; Connor et al., 2024), the goal of which is to map one sample to a transformed version of itself via an autoencoder equipped with transformation operators in latent space. Empirically, these methods are found to somewhat generalize group transformations (e.g., 3D rotations) to object classes that were not seen during training. Other directions include meta-learning approaches (Finn et al., 2017; Yang and Hu, 2021), feature learning approaches (Bach, 2017; Yang et al., 2024a; Jacot et al., 2025) and dynamic architecture choices (Stanley and Miikkulainen, 2002; Chauhan et al., 2024), which could potentially alleviate the problem of having a fixed, frozen, kernel incapable of adapting to problem symmetries. Indeed, our understanding is that networks cannot learn symmetries because their kernel is defined by their architecture, and cannot adapt to the symmetries of a given dataset. Designing mechanisms to adapt the kernel induced by the network to the symmetries of the problem at stake seems a likely path forward to devise more data-efficient deep networks. In future work, our theory could be extended to these methods, in order to establish whether they are theoretically able to learn symmetries from data, and if so, to what extent.

Our theory could also be extended in various ways to capture a richer phenomenology. First, we have mainly focused here on the simple discrete one-dimensional cyclic group. However, our theoretical framework is readily extendable to any finite group including non-abelian groups (see Theorem 15), and it would be interesting to study the practical implications of the theory for such more complex groups. It would also be interesting to understand whether and how the theory could be extended to continuous groups, such as $SO(3)$. Second, our study focuses on symmetries that are native to the space in which the datasets reside (e.g., image rotations). However, group actions may more realistically exist in a latent space affecting the dataset indirectly (e.g., images of 3D-rotated objects). This latter case is particularly interesting because architectures cannot easily be designed to be equivariant to symmetries which are not directly acting in dataset space. We note, however, that given the failure of conventional architectures to correctly generalize simple symmetries native to the dataset space, we do not expect these same architectures to be able to learn more complex, latent space symmetries.

A source of inspiration for learning symmetries could be found in cognitive science and neuroscience. Empirically, there is evidence that humans are superior to deep networks at some tasks necessitating symmetry invariance. In a recent example, Ollikka et al. (2025) showed that humans beat state-of-the-art deep networks and most vision-language models at recognizing objects in unusual poses. Humans are also known to be able to reason about problem symmetries, for example in the problem of mental rotation (Shepard and Metzler, 1971). Interestingly, the time for subjects to compare one 3D object to another in mental rotation tasks is proportional to the angular difference between the two objects, implying that some recurrent processes in the brain may be involved, recurrence lacking from current state-of-the art architectures in deep learning. There is also evidence that the mental rotation ability is learned through experience (or at least not fully operational at birth), as studies on infants show that there is a critical age when they are able to perform this task (Bambha et al., 2022). Conversely, children learning to read need to first unlearn mirror symmetries in order to differentiate characters from their mirrored version (Perea et al., 2011; Pegado et al., 2011). The ability to capture specific problem symmetries may, however, be partially or fully innate in some animals. A study on chicks show that they have the ability to recognize an unknown synthetic 3D object from birth, across multiple points of view (Wood, 2013). This finding prompts us to reconsider whether humans and animals actively learn problem symmetries, or rely on neural architectures which are pre-configured for specific symmetries. It is at least clear from behavioral experiments that humans have the ability to adapt to unnatural symmetric transformations of their environment. Classically, Kohler and Erismann (Kohler, 1963) showed that a subject wearing goggles reversing the world upside down can adapt to this new environment after a few days of practice. The extent to which this adaptation is possible for new, previously unseen symmetries is unclear, however (e.g., what if the goggles permanently permuted the location of every pixel?). Finally, a direction of interest may be to imitate the physicist: by describing the world symbolically, the physicist comes to discover the symmetries of the world and exploit them for predictions. Language models and other neuro-symbolic approaches may be able to partly replicate this ability, or at least piggy-back on human accumulated knowledge about the symmetries of the world.

Acknowledgments

We thank David Radnell, Milo Orlich, Samuel A. Ocko, Jaakko Lehtinen, Tycho F. A. van der Ouderaa, Arthur Jacot, Jan Gerken, Stefano Sarao Mannelli for useful discussions and additional references, Luigi Acerbi and Martin Trapp for comments on the manuscript, Adityanarayanan Radhakrishnan for his clear and concise lecture notes on neural kernels, and Fabio Anselmi for an insight that led to the proof of Theorem 15. We thank the anonymous reviewers and our editor Aapo Hyvärinen for their constructive feedback and suggestions, and for their time. Funding: Research Council of Finland grant to S.D. under the Project “Neuroscience-inspired Deep Learning”: 3357590.

Appendix

Contents

All code is available at `https://github.com/Andrea-Perin/gpsymm`.

Appendix A. A (Very Brief) Introduction to Neural Kernel Theories

Here we briefly introduce the mathematical connection that exists between deep neural networks and kernel methods. We reuse and modify the words of Jan Gerken (`https://gapindnns.github.io/_pages/wide_nns.html`), with permission:

> *The number of neurons in each layer of a neural network is called the* width N *of that layer. When considering the limit in which the width of all hidden layers goes to infinity (assuming that the weights of the network scale as* $1/\sqrt{N}$ *to avoid activity divergence), the neural network simplifies dramatically. By an argument using the central limit theorem, one can show that in the infinite width limit, the neurons follow a collective Gaussian distribution known as a Gaussian process. Intuitively, the fluctuations from all the neurons cancel out. This effect was*

> *known for a long time as the* Neural Network Gaussian Process (NNGP) *theory (Neal, 1996) and characterizes the neural network at initialization, i.e. before training has begun. Furthermore, the neural kernel—characterizing dot product similarity of data points in the last infinitely-wide layer of the network—becomes independent of the specific initialization chosen and can be computed using recursive layer-by-layer expressions.*
>
> *In 2018, a seminal paper by Jacot et al. (2018) showed that the simplifications go even further than that: In the infinite width limit (again assuming that the weights scale as $1/\sqrt{N}$), the dynamics remain manageable even during training. Specifically, they proved that the kernel of an infinitely wide network remains constant throughout training. This kernel, called the* Neural Tangent Kernel (NTK)*, is equal to the NNGP kernel with additional corrective terms. Briefly, this simplification is possible because all weights in the inner layers remain in the vicinity of their initial values during training, allowing to linearize the network around the initial condition. Under the simple but realistic training paradigm of gradient descent of the mean-squared-error loss, the training dynamics can then be solved analytically in closed form and the prediction of the trained network on any input be computed. The output of the trained network is again a Gaussian process.*
>
> *These simplifications in the infinite width limit give powerful insights into the behavior of neural networks.*

Appendix B. Derivation of the Spectral Error

In this section, we report the computations that give the spectral error in Eq. 1. We start by describing general Gaussian process (GP) regression, and then move to the actual derivation of our formula, that applies specifically to a group-cyclic dataset.

We consider a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}$, so that the samples x_i are d dimensional vectors and labels y_i are scalars.

GP regression is a Bayesian method which, starting from the user's prior information (namely, a mean function $m : \mathcal{X} \to \mathbb{R}$, and a covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$), produces a distribution of regression functions of the type $f : \mathcal{X} \to \mathbb{R}$. We are interested in the value of such a function at a given test point $x_t \in \mathcal{X}$, conditioned on the values (x_i, y_i) contained in the dataset. To obtain these pointwise results, one can focus on a finite collection of points $(x_i, y_i) \in \mathcal{X} \times \mathbb{R}$. Any such finite collection has the property (inherited by the GP) of being jointly Gaussian-distributed as follows:

$$(y_1, \ldots, y_N) \sim \mathcal{N}(\vec{m}, K),$$

where $\vec{m}_i = m(x_i) \in \mathbb{R}$ is the mean vector, obtained by evaluating the mean function m over the given points, and $K_{ij} = k(x_i, x_j) \in \mathbb{R}$ is the covariance matrix, obtained by evaluating the covariance function k over all pairs of points x_i, x_j that can be formed in the collection. We set $m(x) = 0$ everywhere.

Equipped with these tools, we can formulate regression on an additional test point x_t as a *conditioning* over a $N+1$ dimensional multivariate Gaussian distribution (MVG), in order to obtain a probability distribution for the value of y_t. In other words, we can consider the set of $N+1$ samples (N from the training set, and the additional one being the test value) to be distributed according to a $N+1$ dimensional MVG:

$$(y_1, \ldots, y_N, y_t) \sim \mathcal{N}\left(0, \begin{bmatrix} K & k(\vec{x}, x_t) \\ k(\vec{x}, x_t)^T & k(x_t, x_t) \end{bmatrix}\right),$$

where we use the notation $k(\vec{x}, x_t) \in \mathbb{R}^N$ to denote the column vector that, at i-th entry, contains $k(x_i, x_t)$ with x_i being the i-th sample in the dataset. The conditional mean and variance for y_t are described by the formulas:

$$\begin{aligned} \mu_{t|\mathcal{D}} &= (K^{-1}k(\vec{x}, x_t))^T \vec{y}, \\ \sigma_{t|\mathcal{D}} &= k(x_t, x_t) - (K^{-1}k(\vec{x}, x_t))^T k(\vec{x}, x_t), \end{aligned}$$

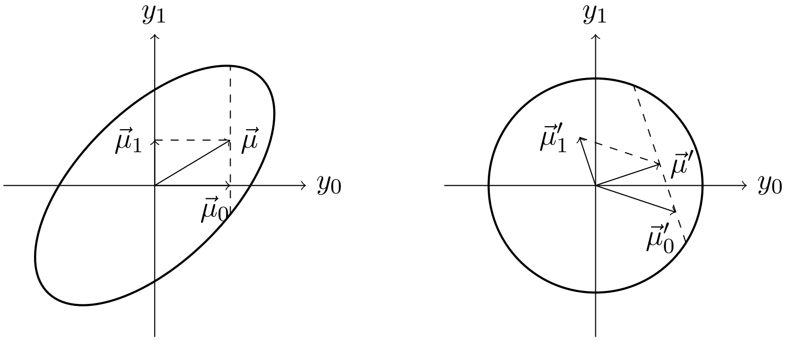
where we define $\vec{y}$ to be the vector of the training labels. The expectation of this distribution (i.e., $\mu_{t|\mathcal{D}}$) will then serve as the result of the Gaussian process regression. This procedure can be extended for p test points, with all the opportune dimensionality changes (i.e., the result of the conditioning will be a p dimensional MVG).

In Section 4.3, we take the Gaussian (RBF) kernel as the kernel for Gaussian Process regression. In later sections, we consider various deep network architectures in the infinite width limit. These also behave like Gaussian Processes, with a kernel that is specified by their architecture (Neal, 1996; Jacot et al., 2018).

Our proof for the spectral formula of the generalization error does not depend on the specific kernel used, as long as the kernel (Gram) matrix is *circulant over a group of points generated by the action of a cyclic group*. We demonstrate elsewhere that this is the case both for the Gaussian kernel, as well as all the deep neural kernels that we analyze in this paper (MLP and ConvNets).

We are now ready to present the derivation of Eq. 1. This equation describes the *most likely* error in prediction resulting from Gaussian Process regression on a dataset generated by cyclic group action. We refer to this *most likely* error as *the result of kernel regression* in main text. We do not study the fluctuations of errors around this mean.

We start by presenting a depiction of the conditioning procedure for the case in which the training set consists of a single point (Fig. 8a). In the following, we denote labels as y_0 and y_1, and their respective values as μ_0 and μ_1. In practice, the ys are the training set labels, and the μs are their values. In this case, the training label is y_0, while the test label is y_1. The two labels are jointly distributed according to a 2 dimensional Gaussian, with mean 0 and covariance matrix K. This matrix depends on the values of x_0, x_1, and the kernel function k. We represent the distribution as an ellipse, which is understood as a level set of the covariance matrix K, centered at the mean, 0, in a way similar to the usual representation of confidence levels of a multivariate Gaussian distribution. Conditioning over the known training label y_0 can be interpreted geometrically as slicing the ellipse with a vertical line at coordinate μ_0. We thus obtain a segment (i.e., a 1-dimensional ellipse), the midpoint of which is the point $(\mu_0, \mu_{1|0})$, where $\mu_{1|0}$ denotes the conditional value of y_1 given x_0, x_1 and the value of y_0. As a consequence, the result of GP regression can thus be interpreted geometrically as finding the y_1 coordinate of the center of the slice. Doing this will reward us with Eq. 1.



(a) Original space.

(b) Whitened space.

Figure 8: 2D sketch of the geometric interpretation, showing the effect of the whitening transformation.

We can make this geometric task easier by turning the ellipse into a circle; we can do this by performing a *whitening* operation on this vector space. Such whitening is effectively obtained by the inverse of the square root of the covariance matrix K. Denoting the transformed values by priming them, our task is now to find $\vec{\mu}'_1$ given $\vec{\mu}'_0$.[10] It is clear that the segment produced by the conditioning turns now into a chord of the circle. This makes the whitened vector $\vec{\mu}'$, which is defined so that it points to the midpoint of the segment, *perpendicular* to the segment itself. This means we can write the following geometric condition:

$$(\vec{\mu}'_0 + \vec{\mu}'_1) \cdot \vec{\mu}'_1 = 0. \tag{3}$$

The same properties hold in the case where we have $2N$ samples, as in the main text, and we condition on $2N - 1$ of them. Let us now denote the value we wish to infer as μ_0; the multidimensional extension of Eq. 3 becomes

$$(\sum_{i=0}^{2N-1} \vec{\mu}'_i) \cdot \vec{\mu}'_0 = 0. \tag{4}$$

We now move to solve Eq. 4. To do so, we make the justified assumption that K is *circulant* (as a consequence of the group-symmetric structure of the dataset). As such, it is diagonalized by the Discrete Fourier Transform (DFT) basis, E:

$$K = E^{-1} \Lambda E.$$

Whitening is obtained by applying the transform given by:

$$Z = K^{-1/2} = E \Lambda^{-1/2} E^{-1},$$

[10] We have now shifted to a vector notation for these values as, unlike in the original space, both components can be nonzero.

to all the vectors $\vec{\mu_i}$.

To make computations easier, we collect our vectors in a data matrix M so that the i-th column is $\vec{\mu}_i$, we can collectively transform our samples with a matrix operation. The collection of transformed (i.e., whitened) samples is denoted as M', and

$$M' = ZM.$$

We now write down a few properties of the matrices at play that will make computations quicker. Denoting δ as the Kronecker delta, λ_i as the i-th eigenvalue of K, and $\omega = \exp(-2\pi i/(2N))$, we have

$$\begin{aligned}
M_{ij} &= \mu_i \delta_{ij}, \\
E_{ij} &= \frac{1}{\sqrt{2N}} \omega^{ij}, \\
E^{-1}_{ij} &= \frac{1}{\sqrt{2N}} \omega^{-ij}, \\
\Lambda^{-1/2}_{ij} &= \frac{1}{\sqrt{\lambda_i}} \delta_{ij}.
\end{aligned}$$

We can now compute the value of the j-th component of the i-th sample in whitened space, that is, M'_{ji}:

$$\begin{aligned}
M'_{ji} &= (ZM)_{ji} \\
&= \sum_k Z_{jk} M_{ki} \\
&= \sum_k (E\Lambda^{-1}E^{-1})_{jk} M_{ki} \\
&= \sum_{klm} E_{jl} \Lambda^{-1/2}_{lm} E^{-1}_{mk} M_{ki} \\
&= \sum_{klm} \frac{1}{\sqrt{2N}} \omega^{jl} \frac{1}{\sqrt{\lambda_l}} \delta_{lm} \frac{1}{\sqrt{2N}} \omega^{-mk} \mu_k \delta_{ki} \\
&= \sum_l \frac{1}{2N} \omega^{jl} \frac{1}{\sqrt{\lambda_l}} \omega^{-li} \mu_i \\
&= \sum_l \frac{1}{2N} \frac{\mu_i \omega^{l(j-i)}}{\sqrt{\lambda_l}}.
\end{aligned}$$

We can now take Eq. 4 and distribute the dot product over the sum. We obtain:

$$\sum_{i=0}^{2N-1} \sum_j M'_{ji} M'_{j0} = 0. \tag{5}$$

In general,

$$\begin{aligned}\sum_j M'_{ji}M'_{j0} &= \sum_j(\sum_l \frac{1}{2N}\frac{\mu_i\omega^{l(j-i)}}{\sqrt{\lambda_l}})(\sum_k \frac{1}{2N}\frac{\mu_0\omega^{kj}}{\sqrt{\lambda_k}}) \\ &= \sum_{lkj} \frac{1}{(2N)^2}\mu_i\mu_0\frac{\omega^{l(j-i)}\omega^{kj}}{\sqrt{\lambda_l\lambda_k}} \\ &= \sum_{lkj} \frac{1}{2N}\frac{\mu_i\mu_0\omega^{-il}}{\sqrt{\lambda_l\lambda_k}}\frac{1}{2N}\omega^{j(l+k)}.\end{aligned}$$

We isolate the rightmost term and use the equality

$$\sum_j \frac{1}{2N}\omega^{j(l+k)} = \delta_{l,2N-k}.$$

Thus, we get

$$\begin{aligned}\sum_j M'_{ji}M'_{j0} &= \sum_{kl} \frac{1}{2N}\frac{\mu_i\mu_0\omega^{-il}}{\sqrt{\lambda_l\lambda_k}}\delta_{l,2N-k} \\ &= \sum_k \frac{1}{2N}\frac{\mu_i\mu_0\omega^{-i(2N-k)}}{\sqrt{\lambda_{2N-k}\lambda_k}} \\ &= \sum_k \frac{1}{2N}\frac{\mu_i\mu_0\omega^{ik}}{\lambda_k}.\end{aligned}$$

Thus, Eq. 5 becomes, after simplifying the common term μ_0 away,

$$\sum_{i=0}^{2N-1}\sum_k \frac{1}{2N}\frac{\mu_i\omega^{ik}}{\lambda_k} = 0.$$

So far, we have only used the circularity of K in our derivation. We now make an additional assumption, this time on the values and ordering of the labels. We require $\mu_{2i} = \mu$, and $\mu_{2i+1} = -\mu$ for $i \in [0, \dots, N-1]$. This amounts to having our samples be interleaved and have opposing labels. We can then write the general expression for the value of μ_i, valid for all i:

$$\mu_i = \mu\omega^{iN} + (\mu_0 - \mu)\delta_{i0}. \tag{6}$$

Note how, for $i = 0$ (that is, the missing point) we recover the unknown value μ_0 for which we want to solve.

With this substitution, we get

$$\sum_{i=0}^{2N-1} \frac{1}{2N}(\mu\omega^{iN} + (\mu_0 - \mu)\delta_{i0})\sum_k \frac{\omega^{ik}}{\lambda_k} = 0.$$

The summation thus splits in three parts:

$$\sum_{i=0}^{2N-1} \frac{\mu}{N} \sum_k \frac{\omega^{i(k+N)}}{\lambda_k} + \sum_{i=0}^{2N-1} \frac{\mu_0}{2N} \sum_k \frac{\omega^{ik}}{\lambda_k}\delta_{i0} - \sum_{i=0}^{2N-1} \frac{\mu}{2N}\delta_{i0} \sum_k \frac{\omega^{ik}}{\lambda_k} = 0$$

We now go over each of them.

- For the first term,

$$\begin{aligned} \sum_{i=0}^{2N-1} \sum_k \frac{\mu}{2N} \frac{\omega^{i(k+N)}}{\lambda_k} &= \mu \sum_k \frac{1}{\lambda_k} \sum_{i=0}^{2N-1} \frac{1}{2N}\omega^{i(k+N)} \\ &= \mu \sum_k \frac{1}{\lambda_k}\delta_{k,-N} \\ &= \mu \frac{1}{\lambda_N}. \end{aligned}$$

- The second term contains a Kronecker delta that selects the 0-th component in the sum over index i. Thus, we get

$$\mu_0 \sum_k \frac{1}{2N}\frac{1}{\lambda_k} = \mu_0 \langle \lambda^{-1} \rangle,$$

  where the angled brackets denote the average.

- The last term, like the previous one, contains a selection on i due to the Kronecker delta. It becomes

$$\mu \langle \lambda^{-1} \rangle.$$

Putting all together, we get

$$\mu \frac{1}{\lambda_N} + \mu_0 \langle \lambda^{-1} \rangle - \mu \langle \lambda^{-1} \rangle = 0,$$

from which, after rearranging, we get a formula for our missing point as a function of the eigenvalues of K:

$$\mu_0 = \mu \left(1 - \frac{\lambda_N^{-1}}{\langle \lambda^{-1} \rangle} \right). \tag{7}$$

The formula for the spectral error, then, is

$$\varepsilon_s \triangleq \frac{\mu_0 - \mu}{\mu} = \frac{\lambda_N^{-1}}{\langle \lambda^{-1} \rangle}. \tag{8}$$

B.1 Extension to Multiple Missing Points

In the case of multiple missing points, we can find an equivalent of Eq. 8. It requires solving a p dimensional linear system, where p is the number of missing points.

We denote by $[p]$ the set of indexes for the missing points. Then, for the missing point with index $j \in [p]$, we have

$$\sum_{m\in[p]} \mu_m \frac{1}{2N}\sum_l \frac{\omega^{l(j-m)}}{\lambda_l} = \mu\left(\frac{1}{2N}\sum_{n\in[p]}\omega^{nN}\sum_k \frac{\omega^{k(j-n)}}{\lambda_k} - \frac{\omega^{jN}}{\lambda_N}\right).$$

We notice that this equation can be expressed as the following linear system:

$$\sum_{m\in[p]} A_{jm}\mu_m = b_j,$$

where

$$A_{jm} = \frac{1}{2N}\sum_l \frac{\omega^{l(j-m)}}{\lambda_l}, \quad b_j = \mu\left(\frac{1}{2N}\sum_{n\in[p]}\omega^{nN}\sum_k \frac{\omega^{k(j-n)}}{\lambda_k} - \frac{\omega^{jN}}{\lambda_N}\right).$$

One can check that, if we set $[p] = \{0\}$, this formula recovers the expression in Eq. 8.

Appendix C. Extension of the Spectral Error to General Finite Groups

We now present a general formula for the spectral error which holds true for arbitrary finite groups, including non-abelian groups.

We start by re-deriving the error for the simple cyclic group using an algebraic formalism which will be easier to generalize for arbitrary finite groups.

C.1 Alternative Derivation of Spectral Error on Cyclic Groups

We re-derive here the formula for the spectral error in Eq. 1. We start by restating the conditions under which the formula holds.

Assume we have an orbit of the cyclic group C_{2N} (where N is the number of points in one class), a labeling function $y : C_{2N} \to \mathbb{R}$ and a kernel function $k : V \times V \to \mathbb{R}$, where $V \cong \mathbb{R}^n$ is some vector space. Additionally, assume that $\rho : C_{2N} \to GL(V)$ is a representation of C_{2N}, and consider its linear action on elements of V. We assume k to be *stationary* with respect to this action. We can then consider, for a "seed point" $x \in V$ the C_{2N} orbit $\mathcal{O}x$ generated by the action of ρ:

$$\mathcal{O}x = \{\rho(g)x \mid g \in C_{2N}\} = \{\rho(r^i)x \mid i \in [2N]\},$$

where $r \in C_{2N}$ is the generator of the cyclic group. We collect the kernel function's values on pairs of points in $\mathcal{O}x$, and store them in the kernel matrix $K \in \mathbb{R}^{2N\times 2N}$, such that

$$K_{ij} = k(x_i, x_j) = k(\rho(r^i)x, \rho(r^j)x) = k(x, \rho(r^{j-i})x),$$

where the last equality is allowed by the stationarity of the kernel function.

Consider now the situation in which we condition kernel regression on all but one point of the dataset. We assume the labels to follow the formula

$$y_{true}(r^i) = (-1)^i, \quad \text{for} \quad i \in [2N],$$

which produces an alternating labeling scheme. Given the symmetry of the dataset, we can choose the missing point to be the first point (indexed at 0) without loss of generality. We want to quantify the *expected error* of kernel regression on the missing point. We can express the prediction given by kernel regression using in the Fourier basis:

$$y = \mathcal{F}\mu - \varepsilon_s \delta_0 = \mathcal{F}(\mu - \varepsilon_s \hat{\delta}_0),$$

where $\mathcal{F} \in \mathbb{C}^{2N \times 2N}$ is the inverse DFT matrix, $\mu \in \mathbb{R}^{2N}$ is the vector of Fourier coefficients for the labeling function, $\hat{\delta}_0$ is the vector of Fourier coefficients for the indicator at 0, and ε_s is the error on the missing point.

Furthermore, for a stationary kernel over an orbit of the group, the kernel matrix is circulant, which allows us to diagonalize it using the same inverse DFT matrix $\mathcal{F}$ (and its transpose):

$$K = \mathcal{F}\Lambda\mathcal{F}^T.$$

Then, as per usual Gaussian Process (GP) regression, we assume a jointly gaussian distribution for the predictions with covariance given by K:

$$p(y) \propto \exp\left(-y^T K^{-1} y\right).$$

Substituting the Fourier expressions,

$$\begin{aligned} y^T K^{-1} y &= (\mathcal{F}(\mu - \varepsilon_s \hat{\delta}_0))^T \mathcal{F}\Lambda^{-1}\mathcal{F}^T(\mathcal{F}(\mu - \varepsilon_s \hat{\delta}_0)) \\ &= \mu^T \Lambda^{-1} \mu - \varepsilon_s \mu^T \Lambda^{-1} \hat{\delta}_0 - \varepsilon_s \hat{\delta}_0^T \Lambda^{-1} \mu + \varepsilon_s^2 \hat{\delta}_0^T \Lambda^{-1} \hat{\delta}_0. \end{aligned}$$

The maximum likelihood estimate for ε_s is obtained by differentiating this expression with respect to it, and setting the result to 0. We obtain

$$\frac{\partial p(y)}{\partial \varepsilon_s} = 0 \implies \mu^T \Lambda^{-1} \hat{\delta}_0 - \varepsilon_s \hat{\delta}_0^T \Lambda^{-1} \hat{\delta}_0 = 0 \implies \varepsilon_s = \frac{\mu^T \Lambda^{-1} \hat{\delta}_0}{\hat{\delta}_0^T \Lambda^{-1} \hat{\delta}_0}.$$

Now, we can compute $\hat{\delta}_0$ (note that we use the orthogonal scaling for the DFT):

$$[\hat{\delta}_0]_i = [\mathcal{F}^T \delta_0]_i = \frac{1}{\sqrt{2N}} \sum_{j=1}^{2N} \omega^{-ij} \delta_{j0} = \frac{1}{\sqrt{2N}}.$$

The denominator becomes

$$\hat{\delta}_0^T \Lambda^{-1} \hat{\delta}_0 = \sum_{ij} \frac{\delta_{ij}}{2N\lambda_i} = \langle \lambda^{-1} \rangle.$$

As for the numerator, since the labeling function is $(y_{true})_i = (-1)^i$, we have

$$\mu_i = [\mathcal{F}^T y_{true}]_i = \frac{1}{\sqrt{2N}} \sum_{j=1}^{2N} \omega^{-ij}(-1)^j = \frac{1}{\sqrt{2N}} \sum_{j=1}^{2N} \omega^{-ij}\omega^{jN} = \sqrt{2N}\delta_{iN}.$$

Thus, the numerator becomes

$$\mu^T \Lambda^{-1} \hat{\delta}_0 = \sum_{i,j=1}^{2N} \sqrt{2N}\delta_{iN} \frac{\delta_{ij}}{\lambda_i} \frac{1}{\sqrt{2N}} = \lambda_N^{-1}.$$

Thus, the formula for the spectral error becomes

$$\varepsilon_s = \frac{\lambda_N^{-1}}{\langle \lambda^{-1} \rangle}.$$

C.2 Extension to Arbitrary Groups

The ideas behind the derivation are the same as in App. C.1, but require the introduction of additional group-theoretic concepts. For a good practical introduction to those topics in representation theory and noncommutative harmonic analysis, we refer to Chirikjian and Kyatkin (2021).

Let us consider a finite group G of order $|G|$, and representation ρ on a d-dimensional vector space V. For a point $x \in V$, consider the orbit generated by the linear action of the representation ρ:

$$\mathcal{O}_x = \{\rho(g)x \mid g \in G\}.$$

Let us also consider a kernel function $k : V \times V \to \mathbb{R}$ that is *stationary* with respect to the action of G, meaning that, for $x, x' \in V$ and for any $g \in G$,

$$k(x, x') = k(g.x, g.x'),$$

where the dot notation stands for the action of the group G onto elements of V. We compute the associated kernel matrix $K \in \mathbb{R}^{|G|\times|G|}$ on the G-orbit we defined above as follows:

$$K_{ij} = k(x_i, x_j) = k(\rho(g_i)x, \rho(g_j)x).$$

Kernel regression produces predictions on input data points. These predictions are given by the function $y : G \to \mathbb{C}$. Under the assumptions of kernel regression, the collection of the $|G|$ predicted values will be jointly Gaussian distributed:

$$p(y) \propto \exp(-\langle y, K^{-1}y \rangle), \tag{9}$$

where the angled brackets denote an inner product in $\mathbb{C}^{|G|}$.

In the following, we are interested in computing the prediction error of a linear kernel regressor on a single missing point in the orbit at $m \in G$. The predictions, conditioned on all but the missing point $m \in G$, can be written:

$$y(g) = y_{true}(g) + \varepsilon\delta_m(g),$$

where we define the G-Kronecker delta function $\delta_h : G \to \mathbb{C}$ as follows:

$$\delta_h(g) = \begin{cases} 1 & \text{if } g = h, \\ 0 & \text{otherwise.} \end{cases}$$

We now express both terms of $y(g)$ in the *generalized Fourier basis*. We start with the true labels y_{true}:

$$y_{true}(g) = \sum_{\rho \in \hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\hat{y}_{true}(\rho)\rho(g)],$$

where $\hat{G}$ denotes the set of equivalent *unitary* irreps of G, d_ρ is the dimension of irrep ρ, and the matrix functions $\hat{y}_{true}(\rho)$ are the generalized Fourier coefficients for the labeling function y_{true}.[11] We then move to the generalized Fourier expression for the G-Kronecker delta. We start from the inverse Fourier transform formula,

$$\delta_h(g) = \sum_{\rho \in \hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\hat{\delta}_h(\rho)\rho(g)].$$

We compute the Fourier coefficients $\hat{\delta}_h(\rho)$ by projecting the function onto the basis given by the irreps:

$$\hat{\delta}_h(\rho) = \sum_{g \in G} \sqrt{\frac{d_\rho}{|G|}} \delta_h(g)\rho(g^{-1}) = \sqrt{\frac{d_\rho}{|G|}} \rho(h^{-1}).$$

Then,

$$\delta_h(g) = \sum_{\rho \in \hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}\left[\sqrt{\frac{d_\rho}{|G|}} \rho(h^{-1})\rho(g)\right] = \frac{1}{|G|} \sum_{\rho \in \hat{G}} d_\rho \mathrm{Tr}[\rho(h^{-1}g)].$$

Analogously to the cyclic case, the Kronecker delta can be seen as a sum of all irreps in the group. We thus obtain the following expression for y:

$$\begin{aligned} y(g) &= y_{true}(g) + \varepsilon\delta_m(g) \\ &= \sum_{\rho \in \hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\hat{y}_{true}(\rho)\rho(g)] + \varepsilon \frac{1}{|G|} \sum_{\rho \in \hat{G}} d_\rho \mathrm{Tr}[\rho(m^{-1}g)] \\ &= \sum_{\rho \in \hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}\left[\left(\hat{y}_{true}(\rho)\rho(m) + \varepsilon\sqrt{\frac{d_\rho}{|G|}}\rho(e)\right)\rho(m^{-1}g)\right]. \end{aligned} \tag{10}$$

We are looking for the most likely prediction error ε on the missing point, which is given by solving the equation:

$$\frac{\partial p(y)}{\partial \varepsilon} = \frac{\partial \exp(-\langle y, K^{-1}y \rangle)}{\partial \varepsilon} = 0. \tag{11}$$

[11] Note that we use the *orthogonal convention* for the normalization constants.

An intermediate step is computing the inner product $\langle y, K^{-1}y\rangle$. We first need to find an expression for K^{-1}. We start by noticing that, due to the stationarity of the kernel function, any one row of the kernel matrix is a *group function.* Indeed, for group elements $g, h \in G$, we have

$$K_{gh} = k(g.x, h.x) = k(x, g^{-1}h.x) \triangleq \kappa_x(g^{-1}h).$$

Thus, for a fixed element $x \in V$ (which we drop from the notation in the following), we can write a row of the kernel matrix in its Fourier form:

$$K_{gh} = \kappa(g^{-1}h) = \sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\hat{\kappa}(\rho)\rho(g^{-1})\rho(h)].$$

We notice that, due to the symmetry of K, it must hold that

$$\mathrm{Tr}[\hat{\kappa}(\rho)\rho(g^{-1})\rho(h)] = \mathrm{Tr}[\hat{\kappa}(\rho)\rho(h^{-1})\rho(g)],$$

which is only possible when $\hat{\kappa}(\rho)$ is real and symmetric for all ρ.

Before we compute Eq. 11, we prove that the specific missing point m can always be set to be the identity element e, by correctly permuting the labeling of y. To do so, we introduce the shift operator L_m, which acts on group functions $f : G \to \mathbb{C}$ as

$$L_m[f](g) = f(m^{-1}g).$$

Its Fourier expression is

$$L_m[f](g) = \sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\widehat{L_m[f]}(\rho)\rho(g)] = \sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\hat{f}(\rho)\rho(m^{-1}g)].$$

By comparing this formula with Eq. 10, we recognize that

$$y(g) = L_m[y_e](g) = y_e(m^{-1}g),$$

where we introduce the labeling function y_e, which indexes the missing point by the group element e:

$$y_e(g) = \sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}\left[\left(\hat{y}_{true}(\rho)\rho(e) + \varepsilon\sqrt{\frac{d_\rho}{|G|}}\rho(e)\right)\rho(g)\right].$$

Thus, we can always set $m = e$ by shifting the labeling function by m^{-1}. The same shifting must be performed on the labels of the kernel matrix. Indeed, if we look at the m-th row of the kernel matrix and shift all group elements by m^{-1}, we get

$$K_{mh} = k(m.x, h.x) \to k(x, (m^{-1}h).x),$$

which means that, after shifting, what used to be the m-th row is now the first row, and that the elements in the row are also shifted by m^{-1}. We conclude that, without loss of generality, we can set up the problem so that the missing point is at group element e. This is what we will do in the rest of the derivation, denoting

$$\hat{y}_e(\rho) = \hat{y}_{true}(\rho) + \varepsilon\sqrt{\frac{d_\rho}{|G|}}\rho(e). \tag{12}$$

We now prove that we can block-diagonalize the kernel matrix by using a matrix composed of the unitary irreps. Let us define the unitary matrix

$$U_{g,(\rho,a,b)} = \sqrt{\frac{d_\rho}{|G|}}\rho_{ab}(g)$$

to be the matrix whose columns are the values taken by the (matrix elements of the) irreps of the group. We then compute $\tilde{K} = U^*KU$, making use of the Schur orthogonality relations:

$$\begin{aligned}
\tilde{K}_{(\sigma,a,b),(\pi,c,d)} &= \sum_{g,h} U^*_{(\sigma,a,b),g} K_{gh} U_{h,(\pi,c,d)} \\
&= \sum_{g,h} \sqrt{\frac{d_\sigma}{|G|}}\overline{\sigma}_{ab}(g) \left(\sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G|}} \mathrm{Tr}[\hat{\kappa}(\rho)\rho(h^{-1})\rho(g)] \right) \sqrt{\frac{d_\pi}{|G|}}\pi_{cd}(h) \\
&= \sum_{g,h} \sqrt{\frac{d_\sigma}{|G|}}\overline{\sigma}_{ab}(g) \left(\sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G|}} \sum_{ijk} \hat{\kappa}(\rho)_{ij}\rho(h^{-1})_{jk}\rho(g)_{ki} \right) \sqrt{\frac{d_\pi}{|G|}}\pi_{cd}(h) \\
&= \sum_{\rho\in\hat{G}} \sum_{ijk} \sqrt{\frac{d_\rho}{|G|}} \left(\sum_g \sqrt{\frac{d_\sigma}{|G|}}\overline{\sigma}_{ab}(g)\rho(g)_{ki} \right) \hat{\kappa}(\rho)_{ij} \left(\sum_h \sqrt{\frac{d_\pi}{|G|}}\rho(h^{-1})_{jk}\pi_{cd}(h) \right) \\
&= \sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho d_\pi d_\sigma}{|G|^3}} \sum_{ijk} \frac{|G|}{d_\rho}\delta_{\sigma\rho}\delta_{bk}\delta_{ai}\hat{\kappa}(\rho)_{ij}\frac{|G|}{d_\rho}\delta_{\rho\pi}\delta_{cj}\delta_{kd} \\
&= \sqrt{\frac{|G|}{d_\sigma}}\delta_{\sigma\pi}\delta_{bd}\hat{\kappa}(\sigma)_{ac}.
\end{aligned}$$

The two Kronecker deltas highlight the doubly block-diagonal structure of the matrix. Rewriting this equation in its matrix form makes this fact immediately clear:

$$\tilde{K} = \bigoplus_{\sigma\in\hat{G}} \sqrt{\frac{|G|}{d_\sigma}} \bigoplus_{i=1}^{d_\sigma} \hat{\kappa}(\sigma).$$

This fact is useful because the inverse of a block-diagonal matrix is the block-diagonal matrix of the inverses. Then,

$$\tilde{K}^{-1}_{(\sigma,a,b),(\pi,c,d)} = \sqrt{\frac{d_\sigma}{|G|}}\delta_{\sigma\pi}\delta_{bd}\hat{\kappa}^{-1}(\sigma)_{ac}.$$

We can now compute the inner product in Eq. 9. For notational convenience, we temporarily write $\bullet_\star$ instead of $\bullet(\star)$ for quantity $\bullet$ indexed by irrep $\star$.

$$
\begin{aligned}
&\langle y, K^{-1}y\rangle = \sum_{g,h} \overline{y(g)} K^{-1}_{gh} y(h) = \\
&= \sum_{g,h}\sum_{\rho\in\hat{G}}\sqrt{\frac{d_\rho}{|G|}}\mathrm{Tr}[\hat{y}^*_{e,\rho}\rho^*(g)]\sum_{\sigma\in\hat{G}}\sqrt{\frac{d_\sigma^3}{|G|^3}}\mathrm{Tr}[\hat{\kappa}_\sigma^{-1}\sigma(h^{-1})\sigma(g)]\sum_{\pi\in\hat{G}}\sqrt{\frac{d_\pi}{|G|}}\mathrm{Tr}[\hat{y}_{e,\pi}\pi(h)] \\
&= \sum_{\rho,\sigma,\pi}\frac{d_\sigma}{|G|}\sqrt{\frac{d_\rho d_\sigma d_\pi}{|G|^3}}\sum_{g,h}\mathrm{Tr}[\hat{y}^*_{e,\rho}\rho^*(g)]\mathrm{Tr}[\sigma(g)\hat{\kappa}_\sigma^{-1}\sigma(h^{-1})]\mathrm{Tr}[\hat{y}_{e,\pi}\pi(h)] \\
&= \sum_{\rho,\sigma,\pi}\frac{d_\sigma}{|G|}\sqrt{\frac{d_\rho d_\sigma d_\pi}{|G|^3}}\sum_{g,h}\sum_{a,b}[\hat{y}^*_{e,\rho}]_{ab}\rho^*(g)_{ba}\sum_{c,d,f}\sigma(g)_{cd}[\hat{\kappa}_\sigma^{-1}]_{df}\sigma(h^{-1})_{fc}\sum_{i,j}[\hat{y}_{e,\pi}]_{ij}\pi(h)_{ji} \\
&= \sum_{\rho,\sigma,\pi}\frac{d_\sigma}{|G|}\sqrt{\frac{d_\rho d_\sigma d_\pi}{|G|^3}}\sum_{a,b,c,d,f,i,j}\left(\sum_{g}\rho^*(g)_{ba}\sigma(g)_{cd}\sum_{h}\sigma(h^{-1})_{fc}\pi(g)_{ji}\right)[\hat{y}^*_{e,\rho}]_{ab}[\hat{\kappa}_\sigma^{-1}]_{df}[\hat{y}_{e,\pi}]_{ij} \\
&= \sum_{\rho,\sigma,\pi}\frac{d_\sigma}{|G|}\sqrt{\frac{d_\rho d_\sigma d_\pi}{|G|^3}}\sum_{a,b,c,d,f,i,j}\left(\frac{|G|}{d_\rho}\delta_{\rho\sigma}\delta_{bc}\delta_{ad}\frac{|G|}{d_\sigma}\delta_{\pi\sigma}\delta_{fj}\delta_{ci}\right)[\hat{y}^*_{e,\rho}]_{ab}[\hat{\kappa}_\sigma^{-1}]_{df}[\hat{y}_{e,\pi}]_{ij} \\
&= \sum_{\sigma}\sqrt{\frac{d_\sigma}{|G|}}\sum_{a,b,c,d,f,i,j}\delta_{bc}\delta_{ad}\delta_{fj}\delta_{ci}[\hat{y}^*_{e,\sigma}]_{ab}[\hat{\kappa}_\sigma^{-1}]_{df}[\hat{y}_{e,\sigma}]_{ij} \\
&= \sum_{\sigma}\sqrt{\frac{d_\sigma}{|G|}}\sum_{a,b,j}[\hat{y}^*_{e,\sigma}]_{ab}[\hat{\kappa}_\sigma^{-1}]_{aj}[\hat{y}_{e,\sigma}]_{bj} \\
&= \sum_{\sigma}\sqrt{\frac{d_\sigma}{|G|}}\mathrm{Tr}[\hat{y}^*_{e,\sigma}\hat{\kappa}_\sigma^{-1}\hat{y}_{e,\sigma}].
\end{aligned}
$$

We can now compute Eq. 11:

$$
\sum_{\sigma}\sqrt{\frac{d_\sigma}{|G|}}\left(\frac{d_\sigma}{|G|}\varepsilon\mathrm{Tr}[\hat{\kappa}_\sigma^{-1}] + \sqrt{\frac{d_\sigma}{|G|}}\mathrm{Tr}[\sigma^*(e)\hat{\kappa}_\sigma^{-1}\hat{y}_{true,\sigma}]\right) = 0
$$

Now, remembering that $\sigma(e) = \mathrm{Id}_{d_\sigma}$, we get the formula for the spectral error for a general finite group G. We state the result as the following theorem:

Theorem 15 *Let G be a finite group, $y : G \to \mathbb{C}$ a labeling function on elements of a G-orbit, $k : G \times G \to \mathbb{R}$ a G-stationary kernel function that operates on pairs of points belonging to a G-orbit. The prediction error over element $e \in G$ by kernel regression is given by*

$$\varepsilon = -\frac{\sqrt{|G|}\sum_{\sigma\in\hat{G}} d_\sigma \mathrm{Tr}[\hat{\kappa}_\sigma^{-1}\hat{y}^*_{true,\sigma}]}{\sum_{\sigma\in\hat{G}} d_\sigma^{3/2}\mathrm{Tr}[\hat{\kappa}_\sigma^{-1}]}, \tag{13}$$

where $\hat{G}$ denotes the set of all equivalent unitary irreps of G, σ denotes an irrep and d_σ its dimension, $\hat{\kappa}_\sigma^{-1}$ are the Fourier coefficients of the first row of the kernel matrix, and $\hat{y}_{true,\sigma}$ are the Fourier coefficients of the ground-truth labeling function.

Remark The generalized spectral error formula differs from the cyclic one in that the alignment between labeling function and inverse kernel is measured as the Frobenius inner product of their generalized Fourier representations (which are stored in matrix coefficients) instead of a simple scalar product in the cyclic case between the Fourier coefficients of the labeling function and of the Gram matrix.

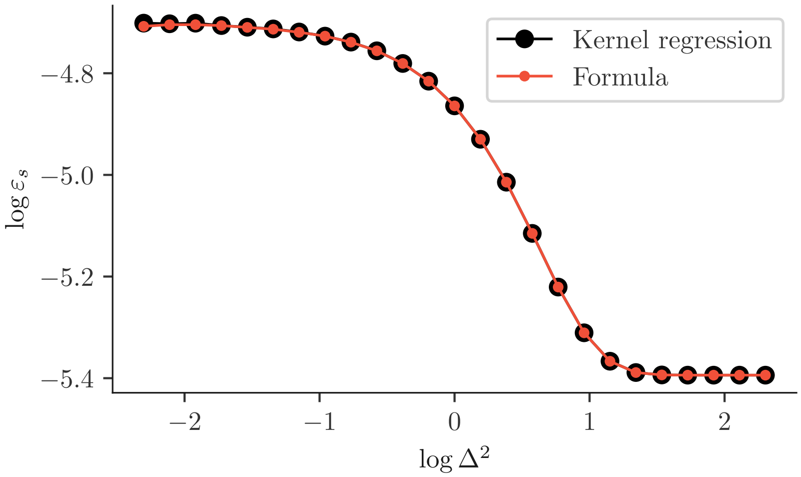


Figure 9: Comparison between the errors as given by Eq. 13 and by standard numerical kernel regression. Used kernel: RBF. Slight discrepancies in the curves are induced by numerical precision errors in the matrix inversion operation required for standard kernel regression.

Experiment In Fig. 9, we include an example of application of the generalized spectral error formula. In this example, the group is the direct product $D_4 \times C_2$, and it is acting on a 2x2 pixel random image to generate the dataset. The action of D_4 is composed of 90 degree rotations and flips of the image, while the C_2 action is a sign flip of all pixel values. The labeling (+1 and -1) follows the sign of the action of the C_2 subgroup. We use the RBF kernel to predict the label on a missing point, as we vary the distance between the positive and the negative portions of the orbit, corresponding to the positive and negative actions of the C_2 subgroup. We compute the prediction error using both our formula and standard kernel regression, and verify that they are equivalent.

Appendix D. Architecture Details

All the relevant code can be found at `https://github.com/Andrea-Perin/gpsymm`.

D.1 Architectures Used in Fig. 1

For Fig. 1, we used the following architectures. We used the default initialization scheme provided by PyTorch (a version of Kaiming uniform).

- a MLP (the `Rearrange` layer is provided by the `einops` library):

```
    m = nn.Sequential(
        Rearrange('b 1 h w -> b (h w)'),
        nn.Linear(1*28*28, 512, bias=True),
        nn.ReLU(),
        nn.Dropout(),
        nn.Linear(512, 128, bias=True),
        nn.ReLU(),
        nn.Dropout(),
        nn.Linear(128, 10, bias=False),
        nn.LogSoftmax(dim=1)
    )
```

- a ConvNet:

```
    c = nn.Sequential(
        nn.Conv2d(1, 24, 5, 1),
        nn.MaxPool2d(kernel_size=2),
        nn.ReLU(),
        nn.Conv2d(24, 32, kernel_size=3),
        nn.MaxPool2d(kernel_size=2),
        nn.ReLU(),
        Rearrange('b c h w -> b (c h w)'),
        nn.Linear(800, 256),
        nn.ReLU(),
        nn.Linear(256, 10),
        nn.LogSoftmax(dim=1)
    )
```

- a ViT-Simple (using the implementation provided by the library `vit-pytorch`):

```
    v = SimpleViT(
        image_size = 28,
        patch_size = 4,
        num_classes = 10,
        dim = 256,
        depth = 2,
        heads = 4,
        mlp_dim = 256,
        channels=1
    )
```

The models were trained for 20 epochs using the Adam optimizer, using a learning rate of 1e-3 and $(\beta_1, \beta_2) = (0.7, 0.9)$. Further details on the implementation can be found in the provided code.

D.2 Architectures Trained on Pairs of Orbits from MNIST

For all tests involving two orbits from MNIST, we used the following MLP architecture, where the parameters `args.n_hidden` defines the depth of the MLP (either 1 or 5).

```
W_std, b_std = 1., 1.
layer = nt.stax.serial(
    nt.stax.Dense(512, W_std=W_std, b_std=b_std),
    nt.stax.Relu(),
)
init_fn, apply_fn, kernel_fn = nt.stax.serial(
    nt.stax.serial(*([layer] * args.n_hidden)),
    nt.stax.Dense(1, W_std=W_std, b_std=b_std)
)
```

For all tests involving two orbits from MNIST, we used the following Convolutional architecture, where the parameter `args.kernel_size` defines the size of the conv kernel, and `IS_GAP` defines whether or not to include the `GlobalAvgPool` layer.

```
W_std, b_std = 1., 1.
conv = nt.stax.serial(
    nt.stax.Conv(
        out_chan=64,
        filter_shape=(args.kernel_size, args.kernel_size),
        padding='CIRCULAR',
        W_std=W_std, b_std=None),
    nt.stax.Relu()
)
pool = nt.stax.GlobalAvgPool() if IS_GAP else nt.stax.Identity()
init_fn, apply_fn, kernel_fn = nt.stax.serial(
    conv,
    pool,
    nt.stax.Flatten(),
    nt.stax.Dense(1, W_std=W_std, b_std=None)
)
```

D.3 Architecture Trained on Multiple Seeds and Multiple Classes of Rotated-MNIST

For Section 4.5, we use the following network architecture and optimizer. Note how this architecture is not in principle trainable with a cross-entropy loss, as it outputs a scalar value. This is needed for the computation of our kernel function. The actual training, which indeed is based on the cross-entropy loss, involves *all but the last two layers*, i.e., the last `Relu` and `Dense` layers. This is done by basic model surgery techniques, made possible by the simple sequential structure of the model.

```
def net_maker(
    W_std: float = 1.,
    b_std: float = 1.,
    dropout_rate: float = 0.5,
    mode: str = ’train’
):
    return nt.stax.serial(
        nt.stax.Dense(512, W_std=W_std, b_std=b_std),
        nt.stax.Relu(),
        nt.stax.Dropout(rate=dropout_rate, mode=mode),
        nt.stax.Dense(128, W_std=W_std, b_std=b_std),
        nt.stax.Relu(),
        nt.stax.Dropout(rate=dropout_rate, mode=mode),
        nt.stax.Dense(10, W_std=W_std, b_std=None),
        nt.stax.Relu(),
        nt.stax.Dense(1, W_std=W_std, b_std=b_std)
    )

optim = optax.adam(learning_rate=1e-3, b1=0.7, b2=0.9)
```

Appendix E. Extension of the Spectral Theory to Equivariant Architectures

We study how the symmetries of the dataset interact with equivariant architectures. We limit our analysis to the Neural Network Gaussian Process (NNGP), as the Neural Tangent Kernel (NTK) adds extra terms that would cloud the derivations. Essentially, the same proofs should hold for the NTK.

We distinguish two cases: (1) when the neural network is equivariant to the symmetry of interest; (2) when the neural network is equivariant, but not to the symmetry of interest. The number of possible symmetries to consider is vast, as well as the number of possible equivariant architectures. Here, we study the interplay between equivariance and symmetries in a limited setup: we consider spatially convolutional neural networks, and we study their interplay with a dataset with translation symmetry (the symmetry to which the network is equivariant) and with a dataset with rotational symmetry. These examples should give the reader an intuition as to how these two things interact in general.

Dataset with translational symmetry. We consider a dataset composed of seed images and all their translations. For the purpose of the proofs to follow, we will focus on a single orbit of this dataset, which consists of a single seed point $x_s \in \mathbb{R}^{n\times n}$ and all of its translations:

$$\mathcal{O}_T = \{g_T^0.x_s, g_T^1.x_s, \cdots g_T^{n-1}.x_s\},$$

where the translation operator g_T acts on images by circularly shifting them along one of the dimensions. For pixel coordinates (i_x, i_y), and the corresponding value of the pixel $x(i_x, i_y)$, we write:

$$g_T.x\left(\begin{bmatrix} i_x \\ i_y \end{bmatrix}\right) = x\left(\begin{bmatrix} (i_x+1) \bmod n \\ i_y \end{bmatrix}\right).$$

Dataset with rotational symmetry. We consider a dataset composed of seed images and all their rotations in C_4 (we limit ourselves to 4 cardinal rotations to avoid definitional problems of image rotation on discrete pixel grids). We will focus on a single orbit of this dataset, which consists in a single seed point $x_s \in \mathbb{R}^{n\times n}$ and all its rotations:

$$\mathcal{O}_R = \{g_R^0.x_s, g_R^1.x_s, g_R^2.x_s, g_R^3.x_s\}$$

where the rotation operator g_R permutes pixel coordinates (i_x, i_y) as follows:

$$g_R.x\left(\begin{bmatrix} i_x \\ i_y \end{bmatrix}\right) = x\left(\begin{bmatrix} i_y \\ n - i_x \end{bmatrix}\right).$$

We also distinguish between two types of convolutional architectures: (1) convolutional architectures where the last layer is fully connected: these architectures do not ensure full invariance to translation; (2) convolutional architectures where the last layer does global average pooling, ensuring invariance to translation.

Fully connected convolutional network (FC). We consider a fully connected convolutional network with one hidden layer, filters of size 3x3, circular padding and stride of 1. The network $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ is parameterized by:

$$f_{\text{FC}}(x) = A\frac{1}{\sqrt{k}}\phi(B \circledast x)_v,$$

where $A \in \mathbb{R}^{1\times n^2 k}$, $B \in \mathbb{R}^{k\times 1\times 3\times 3}$, $\circledast$ denotes the spatial convolution operation, and for any matrix $u \in \mathbb{R}^{n\times n}$, $u_v \in \mathbb{R}^{n^2}$ denotes the vectorization (i.e., flattening) of u. This network first applies a convolutional layer to the data, then flattens the resulting representation into a vector, and passes it through a fully connected layer.

Let $K \in \mathbb{R}^{n^2\times n^2}$ denote the Conv-NNGP for a 1-layer fully convolutional network operating on a pair of images x, x' of size $n \times n$. This kernel contains an entry for every coordinate quadruplet (i_x, i_y, i'_x, i'_y) across the images x, x'. On the other hand, the Conv-NNGP of the network above f_{FC} reduces this representation to a kernel of size $K_{\text{FC}} \in \mathbb{R}$, by letting $K_{\text{FC}} = \text{Tr}(K)$ (Arora et al., 2019):

$$\begin{aligned} K_{\text{FC}}(x, x') &= \sum_{i_x, i_y} K_{i_x, i_y, i_x, i_y}(x, x') \\ &= \sum_{i_x, i_y} \sum_k \Big\langle \phi(B_{k, i_x, i_y} \cdot x), \phi(B_{k, i_x, i_y} \cdot x') \Big\rangle_\Theta \end{aligned}$$

where k denotes the filter index, and $\langle ., . \rangle_\Theta$ the average dot product of the embeddings over the randomly sampled weights of the models.

Global Average Pooling convolutional network (GAP). We consider a convolutional network with global average pooling at the last layer. This network is invariant to discrete translations. We consider the network $f_{\text{GAP}} : \mathbb{R}^{n\times n} \to \mathbb{R}$ be parameterized by:

$$f_{\text{GAP}}(x) = \frac{1}{\sqrt{k}n^2} \sum_k \sum_{i_x} \sum_{i_y} A_{1k}\phi(B_{k,i_x,i_y} \cdot x)$$

where $A \in \mathbb{R}^{1\times k}$, and $B \in \mathbb{R}^{k\times 1\times 3\times 3}$ with $B_k \in \mathbb{R}^{1\times 1\times 3\times 3}$ indexing filter k of B. $B_{k,i_x,i_y} \in \mathbb{R}^{1\times 1\times n\times n}$ is obtained by centering B_k at coordinates (i_x, i_y) of an $n \times n$ grid with periodic boundary conditions, and filling the remaining entries with zeros. Then, the dot product is understood as the sum of the elementwise multiplications of all entries of x and B_{k,i_x,i_y}. We remark that this operation is effectively an alternative way of describing a convolution with filter bank B, but this indexing choice proves useful in the proofs. After applying a convolutional layer to the data, this network averages the resulting representation across each of the k output channels, and then takes a linear combination of these averages using a fully connected layer.

The Conv-NNGP of f_{GAP} reduces K to K_{GAP} by letting $K_{\text{GAP}} = \frac{1}{n^4} \sum_{i_x,i_y,i'_x,i'_y} K_{i_x,i_y,i'_x,i'_y}$ (i.e., averaging over all elements of K). In the following, we omit this prefactor $1/n^4$ for notational convenience.

$$\begin{aligned}
K_{\text{GAP}}(x, x') &= \sum_{\substack{i_x,i_y\\ i'_x,i'_y}} K_{i_x,i_y,i'_x,i'_y}(x, x') \\
&= \sum_{\substack{i_x,i_y\\ i'_x,i'_y}} \sum_k \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i'_x,i'_y} \cdot x') \Big\rangle_\Theta
\end{aligned}$$

Proposition 11 *The kernel matrix of a fully connected convolutional network K_{FC} over a translation orbit O_T is circulant. Moreover, this kernel matrix is in general not constant or rank-deficient.*

Proof

We first show that the kernel of a fully connected network is circulant over an orbit of image rotations. Consider a pair of images from this orbit $(x, x') \in O_T$. We show below that applying the same image rotation g_T to both these images leaves the kernel unaffected, which is equivalent to showing that the kernel matrix is circulant over the orbit O_T. First, we write:

$$\begin{aligned}
K_{\text{FC}}(g_T.x, g_T.x') &= \sum_{i_x,i_y} \sum_k \Big\langle \phi(B_{k,i_x,i_y} \cdot g_T.x), \phi(B_{k,i_x,i_y} \cdot g_T.x) \Big\rangle_\Theta \\
&= \sum_{i_x,i_y} \sum_k \Big\langle \phi(g_T^{-1}.B_{k,i_x,i_y} \cdot x), \phi(g_T^{-1}.B_{k,i_x,i_y} \cdot x) \Big\rangle_\Theta
\end{aligned}$$

Applying a translation to image x is equivalent to applying a change of spatial index to the convolutional filters, $g_T^{-1}.B_{k,i_x,i_y} = B_{k,i_x-1 \bmod n,i_y}$, such that:

$$K_{\text{FC}}(g_T \cdot x, g_T \cdot x') = \sum_{i_x,i_y} \sum_k \Big\langle \phi(B_{k,i_x-1 \bmod n} \cdot x), \phi(B_{k,i_x-1 \bmod n,i_y} \cdot x) \Big\rangle_\Theta$$

We operate the changes of variable with renaming $i_x \leftarrow i_x - 1 \bmod n$:

$$\begin{aligned} K_{\mathrm{FC}}(g_T.x, g_T.x') &= \sum_{i_x, i_y} \sum_k \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i_x,i_y} \cdot x) \Big\rangle_\Theta \\ &= K_{\mathrm{FC}}(x, x') \end{aligned}$$

This concludes our proof that the kernel of a fully connected convolutional network is circulant over an orbit of image translations.

Another question of interest is whether the kernel is constant over pairs of images belonging to the same orbit. Consider an image x and its transformation $g_T.x$.

$$\begin{aligned} K_{\mathrm{FC}}(x, g_T.x) &= \sum_{i_x, i_y} \sum_k \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i_x,i_y} \cdot g_T.x) \Big\rangle_\Theta \\ &= \sum_{i_x, i_y} \sum_k \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(g_T^{-1}.B_{k,i_x,i_y} \cdot x) \Big\rangle_\Theta \\ &= \sum_{i_x, i_y} \sum_k \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i_x-1 \bmod n,i_y} \cdot x) \Big\rangle_\Theta \end{aligned}$$

It is clear that there is no change of variable that could make this kernel equal to $K_{\mathrm{FC}}(x, x)$. In general,

$$K_{\mathrm{FC}}(x, g_T.x) \neq K_{\mathrm{FC}}(x, x)$$

In other words, the fully connected convolutional network does not in general produce invariance to the translation transformation. Its kernel over an orbit is in general not constant.

To prove that K_{FC} is not, in general, rank-deficient, one can consider a seed image where only one pixel is active, and consider shifts in the images of size larger than the filter's size. Doing so results in a kernel matrix over the orbit that is a multiple of the identity, which is full rank.

■

Proposition 12 *The kernel matrix of a global average pooling convolutional network K_{GAP} over a translation orbit O_T is constant.*

Proof

We now study how a global average pooling layer affects the kernel. We consider two successive images from the translation orbit $(x, g_T.x) \in O_T$:

$$\begin{aligned}
K_{\mathrm{GAP}}(x, g_T.x) &= \sum_{\substack{i_x,i_y\\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i'_x,i'_y} \cdot (g_T.x)) \right\rangle_\Theta \\
&= \sum_{\substack{i_x,i_y\\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(g_T^{-1}.B_{k,i'_x,i'_y} \cdot (x)) \right\rangle_\Theta \\
&= \sum_{\substack{i_x,i_y\\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i'_x-1 \bmod n,i'_y} \cdot (x)) \right\rangle_\Theta
\end{aligned}$$

By the change of variable with renaming $i'_x \leftarrow i'_x - i \mod n$, we recover:

$$\begin{aligned}
K_{\mathrm{GAP}}(x, g_T.x) &= \sum_{\substack{i_x,i_y\\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i'_x,i'_y} \cdot x) \right\rangle_\Theta \\
&= K_{\mathrm{GAP}}(x, x)
\end{aligned}$$

The kernel of a global average pooling convolutional network computed over a translation orbit is thus constant.

In our spectral error formula, the kernel over a pair of orbits will thus be rank-deficient (each orbit-wise block being constant), such that some inverse eigenvalues in the denominator will diverge. The spectral error will thus go to 0 (perfect generalization). We recover the well-known fact that a global average pooling convolutional network is invariant to translations.

■

Proposition 13 *The kernel matrix of a fully connected convolutional network K_{FC} over a rotation orbit O_R is circulant, but in general not rank-deficient or constant.*

Proof Now we show that the kernel of a fully connected convolutional network is circulant over an orbit of image rotations. Consider a pair of images from this orbit $(x, x') \in O_R$. We show next that applying the same image rotation g_R to both these images leaves the kernel unaffected:

$$\begin{aligned}
K_{\mathrm{FC}}(g_R.x, g_R.x') &= \sum_{i_x,i_y} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot g_R.x), \phi(B_{k,i_x,i_y} \cdot g_R.x) \right\rangle_\Theta \\
&= \sum_{i_x,i_y} \sum_k \left\langle \phi(g_R^{-1}.B_{k,i_x,i_y} \cdot x), \phi(g_R^{-1}.B_{k,i_x,i_y} \cdot x) \right\rangle_\Theta
\end{aligned}$$

Applying a rotation to image x is equivalent to applying a change of index to the convolutional filters B_{k,i_x,i_y} jointly with rotating the filters. We denote $B'_{k,n-i_y,i_x} = g_R^{-1}.B_{k,i_x,i_y}$.

$$K_{\mathrm{FC}}(g_R \cdot x, g_R \cdot x') = \sum_{i_x,i_y} \sum_k \left\langle \phi(B'_{k,n-i_y,i_x} \cdot x), \phi(B'_{k,n-i_y,i_x} \cdot x) \right\rangle_\Theta$$

We operate the changes of variable with renaming $i_x \leftarrow n - i_y$, $i_y \leftarrow i_x$.

$$K_{\mathrm{FC}}(g_R.x, g_R.x') = \sum_{i_x,i_y} \sum_{k} \Big\langle \phi(B'_{k,i_x,i_y} \cdot x), \phi(B'_{k,i_x,i_y} \cdot x) \Big\rangle_{\Theta}$$

Assuming that the weights Θ of the filters B_k are drawn from a random normal distribution $B_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then $B'_k = g_R^{-1}.B_k$ will have the same distribution of weights $B'_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Thus:

$$K_{\mathrm{FC}}(g_R.x, g_R.x') = K_{\mathrm{FC}}(x, x')$$

This concludes our proof that the kernel of a fully connected convolutional network is circulant over an orbit of image rotations.

Another question of interest is whether the kernel is constant or rank-deficient over pairs of images belonging to the same orbit. Consider an image x and its transformation $g_R.x$.

$$\begin{aligned} K_{\mathrm{FC}}(x, g_R.x) &= \sum_{i_x,i_y} \sum_{k} \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i_x,i_y} \cdot g_R.x) \Big\rangle_{\Theta} \\ &= \sum_{i_x,i_y} \sum_{k} \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(g_R^{-1}.B_{k,i_x,i_y} \cdot x) \Big\rangle_{\Theta} \\ &= \sum_{i_x,i_y} \sum_{k} \Big\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B'_{k,n-i_y,i_x} \cdot x) \Big\rangle_{\Theta} \end{aligned}$$

There is no change of variable that could make this kernel equal to $K_{\mathrm{FC}}(x, x)$. In general,

$$K_{\mathrm{FC}}(x, g_R.x) \neq K_{\mathrm{FC}}(x, x)$$

In other words, the fully connected network does not produce invariance to the rotation transformation. Its kernel over an orbit is in general not constant.

A similar argument as in the case of shifts can be made to prove that the kernel is not, in general, rank-deficient. This can be proven by choosing a “Dirac-like” image whose only active pixel is not at the center of the image. ■

Proposition 14 *The kernel matrix of a global average pooling convolutional network K_{GAP} over a rotation orbit O_R is circulant, but in general not rank-deficient or constant.*

Proof We study how changing the last layer from a fully connected to a global average pooling layer affects the results of the previous proposition.

$$\begin{aligned} K_{\mathrm{GAP}}(g_R.x, g_R.x') &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_{k} \Big\langle \phi(B_{k,i_x,i_y} \cdot g_R.x), \phi(B_{k,i'_x,i'_y} \cdot g_R.x') \Big\rangle_{\Theta} \\ &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_{k} \Big\langle \phi(g_R^{-1}.B_{k,i_x,i_y} \cdot x), \phi(g_R^{-1}.B_{k,i'_x,i'_y} \cdot x') \Big\rangle_{\Theta} \\ &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_{k} \Big\langle \phi(B'_{k,n-i_y,i_x} \cdot x), \phi(B'_{k,n-i'_y,i'_x} \cdot x') \Big\rangle_{\Theta} \end{aligned}$$

where we denote $B'_{k,n-i_y,i_x} = g_R^{-1}.B_{k,i_x,i_y}$ the filter rotated and displaced by the rotation. Changing variables as $i_x \leftarrow n - i_y$, $i_y \leftarrow i_x$, $i'_x \leftarrow n - i'_y$, $i'_y \leftarrow i'_x$, we get

$$\begin{aligned} K_{\text{GAP}}(g_R.x, g_R.x') &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_k \left\langle \phi(B'_{k,i_x,i_y} \cdot x), \phi(B'_{k,i'_x,i'_y} \cdot x') \right\rangle_\Theta \\ &= K_{\text{GAP}}(x, x') \end{aligned}$$

The kernel of this network is thus circulant on a single orbit, as before with the fully connected convolutional network.

We now compute the kernel value for pairs of images belonging to the same orbit, in order to check whether the kernel is constant over an orbit:

$$\begin{aligned} K_{\text{GAP}}(x, g_R.x) &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B_{k,i'_x,i'_y} \cdot g_R.x) \right\rangle_\Theta \\ &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(g_R^{-1}.B_{k,i'_x,i'_y} \cdot x) \right\rangle_\Theta \\ &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B'_{k,n-i'_y,i'_x} \cdot x) \right\rangle_\Theta \end{aligned}$$

Changing variables as $i'_x \leftarrow n - i'_y$, $i'_y \leftarrow i'_x$, we get:

$$\begin{aligned} K_{\text{GAP}}(x, g_R.x) &= \sum_{\substack{i_x,i_y \\ i'_x,i'_y}} \sum_k \left\langle \phi(B_{k,i_x,i_y} \cdot x), \phi(B'_{k,i'_x,i'_y} \cdot x) \right\rangle_\Theta \\ &\neq K_{\text{GAP}}(x, x) \end{aligned}$$

In general, the kernel is not constant over an orbit.

To prove that this kernel over an orbit is in general not rank deficient, one can resort to using a seed image that is the sum of two “Dirac-like” images, each with a single nonzero pixel, and both of them in an area contained within the filter size. ■

Appendix F. Varying the Number of Angles in an Orbit

We report here plots comparing spectral and exact NTK errors for varying number of rotations for a 1 hidden layer MLP.

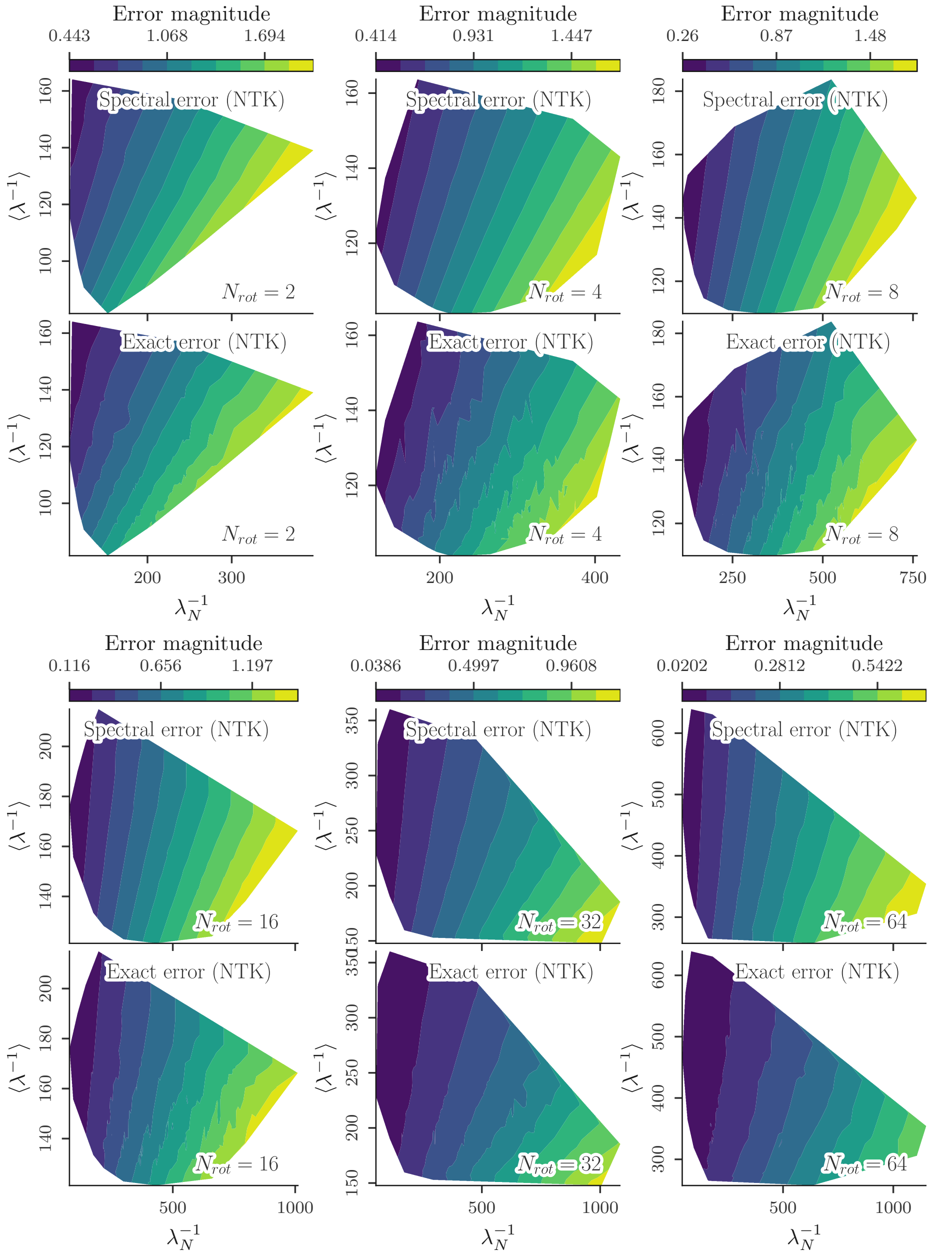


Figure 10: Comparison of spectral and exact NTK errors across a range of values for λ_N and $\langle\lambda^{-1}\rangle$.

Appendix G. Further MLP Analyses

We report here plots of the NTK analysis for a deeper MLP (5 hidden layers, Fig. 11), and a MLP that is trained via Adam (Fig. 12).

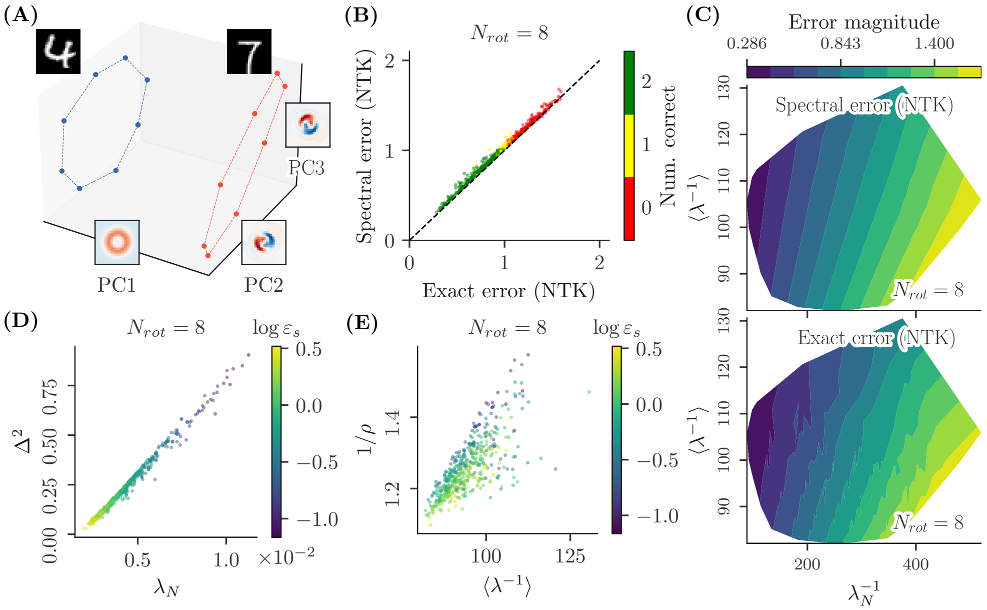


Figure 11: MLP, 5 hidden layers.

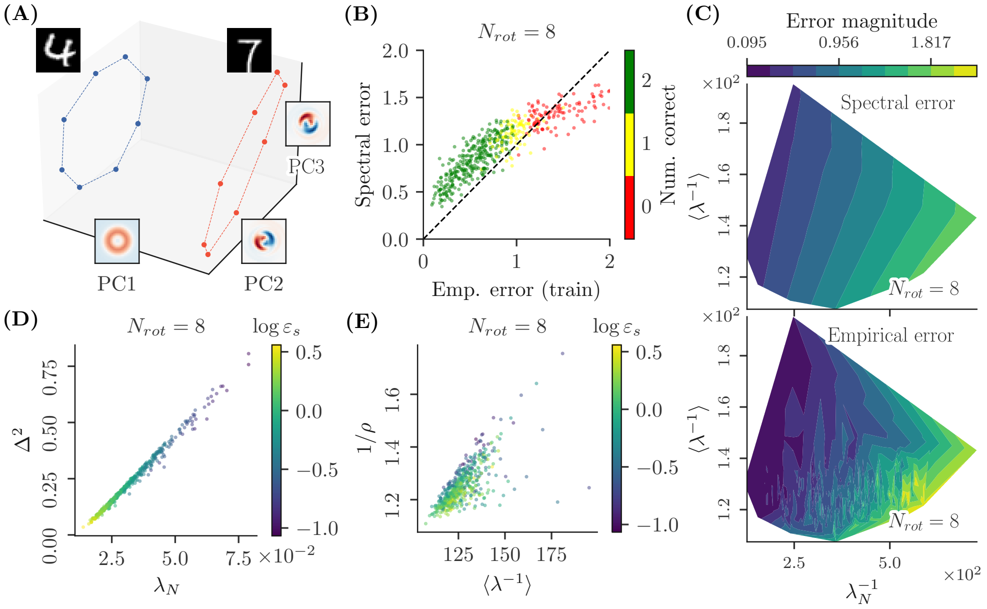


Figure 12: Trained MLP.

Appendix H. Multiple Seeds, Multiple Classes - Additional Figures

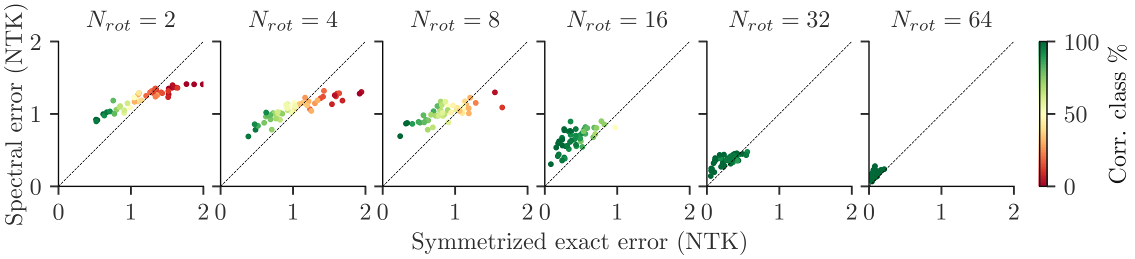


Figure 13: **Comparison of spectral and exact NTK errors for the multiple seeds case.** We compare the spectral error, averaged over all the pairings of orbits for a given number of pairs, and the symmetrized NTK prediction error, over a range of number of rotations in the orbits, N_{rot}. We superimpose the bisector as a visual reference. The color coding reflects the percentage of seeds (of both classes) for which the NTK regression gives a correct prediction, understood as agreeing with the +1 label of the missing points. As the number of points in the orbits increases, both errors decrease as expected, and the percentage of correct NTK predictions increases.

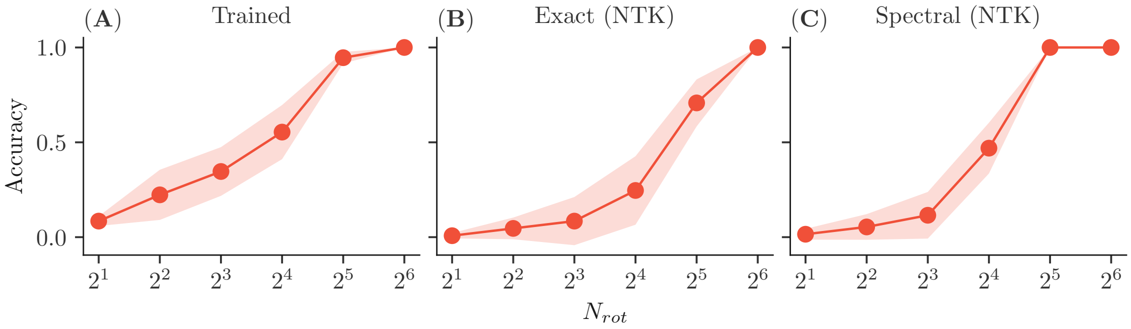


Figure 14: **Our spectral method correctly reproduces the generalization behavior of a trained finite-width MLP on rotated-MNIST, as we vary the number of sampled angles.** On a version of rotated-MNIST comprising all 10 classes and 13 seed images per class, we compare (A) a cross-entropy trained MLP (2 hidden layers) (B) a one-versus-all NTK regression strategy for this same architecture (we exclude one angle from one class and use NTK regression on the missing points, assuming label +1 for this class and label -1 for all other classes, repeat over all leave-out classes and average classification accuracy over all missing points), and (C) our multi-class adapted spectral error (see text). As the number of sampled angles in the orbits increases, the accuracies of all methods increase similarly and gradually on the missing points, showing that no mechanism for symmetry learning is present for the finite-width network that would be missing from our spectral theory, or from simple NTK regression.

Appendix I. Further Convolutional NTK (CNTK) Analyses

I.1 Rotation Orbits

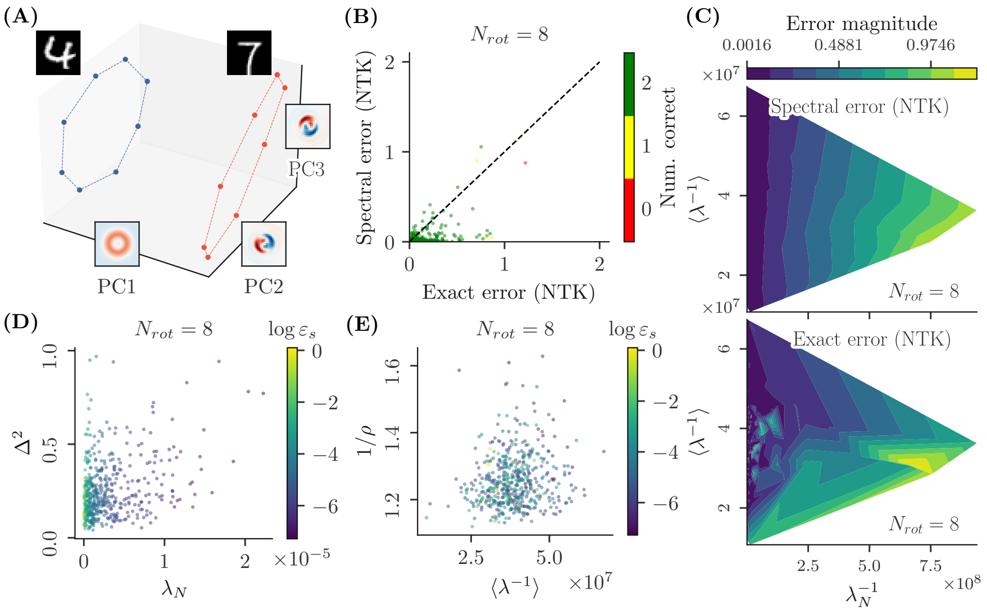


Figure 15: CNTK, Global Average Pooling, on rotation orbit.

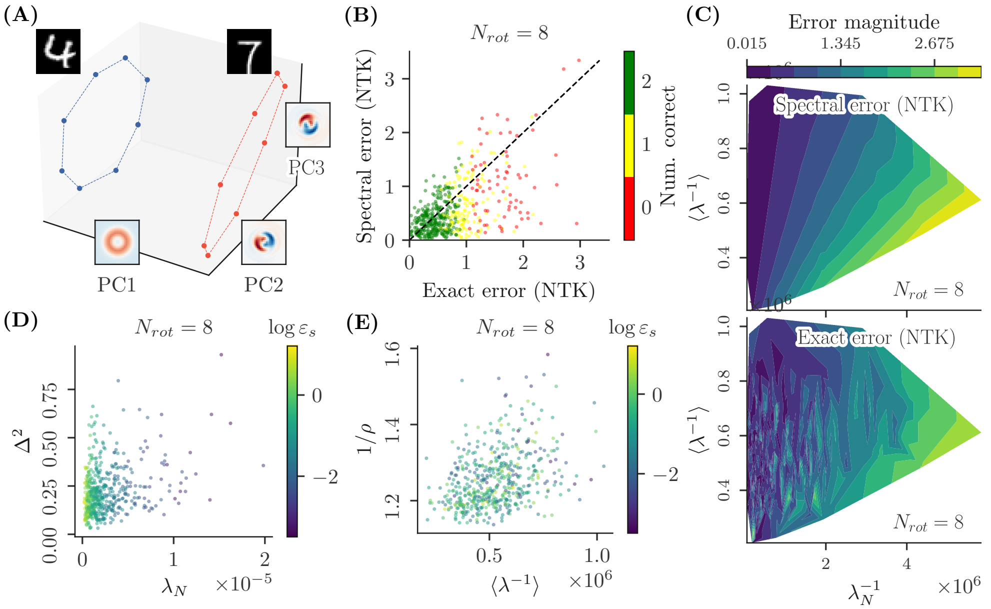


Figure 16: CNTK, Global Average Pooling, on rotation orbit. Kernel size (4, 4), strides (4, 4).

I.2 Translation Orbits

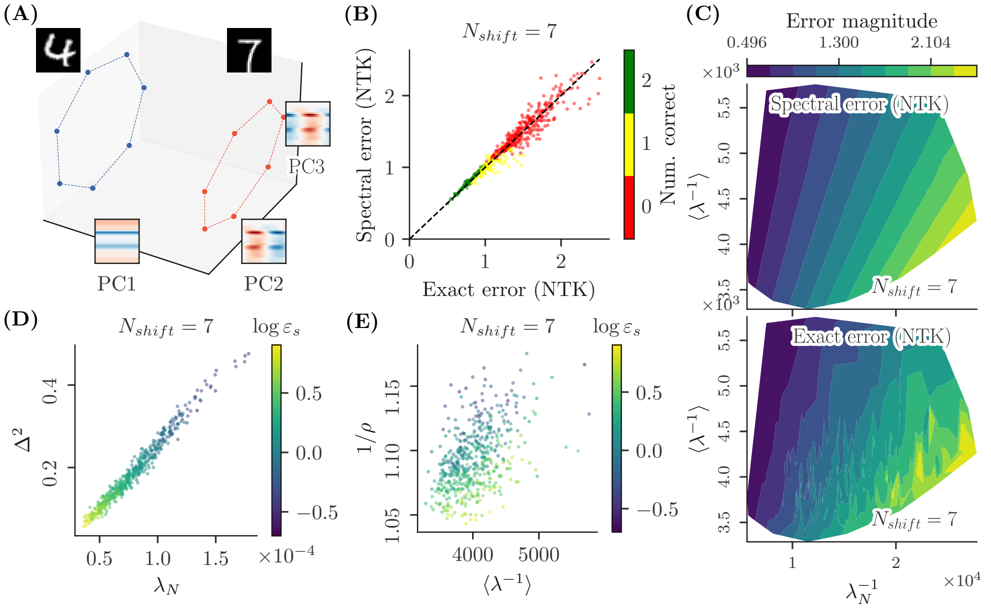


Figure 17: CNTK, Fully Connected, on translation orbit.

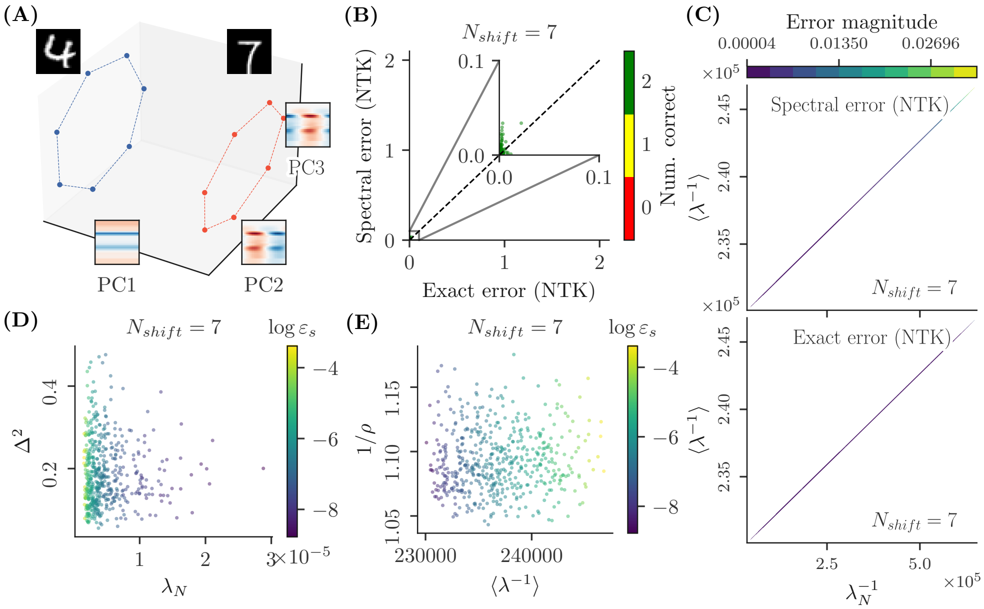


Figure 18: CNTK, Global Average Pooling, on translation orbit.

References

Amro Abbas and Stéphane Deny. Progress and limitations of deep networks to recognize objects in unusual poses. In *Conference on Artificial Intelligence (AAAI)*, 2023.

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations (ICLR)*, 2023.

Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Fabio Anselmi, Georgios Evangelopoulos, Lorenzo Rosasco, and Tomaso Poggio. Symmetry-adapted representation learning. *Pattern Recognition*, 2019.

Fabio Anselmi, Luca Manzoni, Alberto d'Onofrio, Alex Rodriguez, Giulio Caravagna, Luca Bortolussi, and Francesca Cairoli. Data symmetries and learning in fully connected neural networks. *IEEE Access*, 2023.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Yehonatan Avidan, Qianyi Li, and Haim Sompolinsky. Unified theoretical framework for wide neural network learning dynamics. *Physical Review E*, 2025.

Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research (JMLR)*, 2019.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research (JMLR)*, 2017.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences (PNAS)*, 2024.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arXiv preprint*, 2023.

Valerie Bambha, Aaron Beckner, Nikita Shetty, Annika Voss, Jinlin Xie, Eunice Yiu, Vanessa LoBue, Lisa Oakes, and Marianella Casasola. Developmental changes in children's object insertions during play. *Journal of Cognition and Development*, 2022.

Erik J Bekkers. B-spline CNNs on Lie groups. *arXiv preprint*, 2021.

Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew G Wilson. Learning invariances in neural networks from training data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning (ICML)*, 2020.

Diane Bouchacourt, Mark Ibrahim, and Stéphane Deny. Addressing the topological defects of disentanglement via distributed operators. *arXiv preprint*, 2021.

Johann Brehmer, Sönke Behrends, Pim de Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv preprint*, 2024.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint*, 2021.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 2021.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A. Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 2024.

Di Chen, Yiwei Bai, Wenting Zhao, Sebastian Ament, John Gregoire, and Carla Gomes. Deep reasoning networks for unsupervised pattern de-mixing with constraint reasoning. In *International Conference on Machine Learning (ICML)*, 2020.

Gregory S. Chirikjian and Alexander B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis.* CRC Press, 2021.

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

SueYeon Chung, Daniel D. Lee, and Haim Sompolinsky. Classification and geometry of general perceptual manifolds. *Physical Review X*, 2018.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2016.

Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *International Conference on Machine Learning (ICML)*, 2019a.

Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant CNNs on homogeneous spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

Uri Cohen, SueYeon Chung, Daniel D. Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature Communications*, 2020.

Marissa Connor and Christopher Rozell. Representing closed transformation paths in encoded network latent space. *Conference on Artificial Intelligence (AAAI)*, 2020.

Marissa Connor, Bruno Olshausen, and Christopher Rozell. Learning internal representations of 3D transformations from 2D projected inputs. *Neural Computation*, 2024.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Benjamin Culpepper and Bruno Olshausen. Learning transport operators for image manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

Stéphane D'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning (ICML)*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Emilien Dupont, Miguel Bautista Martin, Alex Colburn, Aditya Sankar, Josh Susskind, and Qi Shan. Equivariant neural rendering. In *International Conference on Machine Learning (ICML)*, 2020.

Gamaleldin Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. Revisiting spatial invariance with low-rank local connectivity. In *International Conference on Machine Learning (ICML)*, 2020.

Babak Esmaeili, Robin Walters, Heiko Zimmermann, and Jan-Willem van de Meent. Topological obstructions and how to avoid them. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Matthew Farrell, Blake Bordelon, Shubhendu Trivedi, and Cengiz Pehlevan. Capacity of group-invariant linear readouts from equivariant representations: How many objects can be linearly classified under all possible views? In *International Conference on Learning Representations (ICLR)*, 2022.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to Lie groups on arbitrary continuous data. In *International Conference on Machine Learning (ICML)*, 2020.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: An empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. Learning and leveraging world models in visual representation learning. *arXiv preprint*, 2024.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020.

Robert Gens and Pedro Domingos. Deep symmetry networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Federica Gerace, Luca Saglietti, Stefano Sarao Mannelli, Andrew Saxe, and Lenka Zdeborová. Probing transfer learning with a model of synthetic correlated datasets. *Machine Learning: Science and Technology*, 2022.

Jan E Gerken and Pan Kessel. Emergent equivariance in deep ensembles. In *International Conference on Machine Learning (ICML)*, 2024.

Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 2020.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

David J. Gross. The role of symmetry in fundamental physics. *Proceedings of the National Academy of Sciences (PNAS)*, 1996.

Nate Gruver, Marc Anton Finzi, Micah Goldblum, and Andrew Gordon Wilson. The Lie derivative for measuring learned equivariance. In *International Conference on Learning Representations (ICLR)*, 2023.

Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological deep learning: Going beyond graph data. *arXiv preprint*, 2023.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint*, 2018.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep networks always grok and here is why. In *International Conference on Machine Learning (ICML)*, 2024.

Mark Ibrahim, Diane Bouchacourt, and Ari Morcos. Robust self-supervised learning with Lie groups. *arXiv preprint*, 2022.

Mark Ibrahim, Quentin Garrido, Ari S. Morcos, and Diane Bouchacourt. The robustness limits of soTA vision models to natural variation. *Transactions on Machine Learning Research (TMLR)*, 2023.

Alexander Immer, Tycho F.A. van der Ouderaa, Gunnar Rätsch, Vincent Fortuin, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable laplace approximations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Arthur Jacot, Seok Hoan Choi, and Yuxiao Wen. How DNNs break the curse of dimensionality: Compositionality and symmetry learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Sékou-Oumar Kaba and Siamak Ravanbakhsh. Symmetry breaking and equivariant neural networks. In *NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint*, 2020.

T. Anderson Keller and Max Welling. Topographic vaes learn equivariant capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

T. Anderson Keller and Max Welling. Topographic vaes learn equivariant capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021b.

Ivo Kohler. The formation and transformation of the perceptual world. *Psychological issues*, 1963.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations (ICLR)*, 2018.

Qiyao Liang, Ziming Liu, Mitchell Ostrow, and Ila Fiete. How diffusion models learn to factorize and compose. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Samuel Lippl and Kim Stachenfeld. When does compositional structure yield compositional generalization? A kernel theory. In *International Conference on Learning Representations (ICLR)*, 2025.

Spandan Madan, Timothy Henry, Jamell Dozier, Helen Ho, Nishchal Bhandari, Tomotake Sasaki, Frédo Durand, Hanspeter Pfister, and Xavier Boix. When and how convolutional neural networks generalize to out-of-distribution category–viewpoint combinations. *Nature Machine Intelligence*, 2022.

Spandan Madan, Tomotake Sasaki, Hanspeter Pfister, Tzu-Mao Li, and Xavier Boix. In-distribution adversarial attacks on object recognition models using gradient-free search. *arXiv preprint*, 2025.

Giovanni Luca Marchetti, Christopher J Hillar, Danica Kragic, and Sophia Sanborn. Harmonics of learning: Universal fourier features emerge in invariant networks. In *Conference on Learning Theory (COLT)*, 2024.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2018.

Gabriel Mel and Surya Ganguli. A theory of high dimensional regression with arbitrary correlations between input features and target functions: Sample complexity, multiple descent curves and a hierarchy of phase transitions. In *International Conference on Machine Learning (ICML)*, 2021.

Giangiacomo Mercatali, Andre Freitas, and Vikas Garg. Symmetry-induced disentanglement on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Artem Moskalev, Anna Sepliarskaia, Erik J Bekkers, and Arnold W.M. Smeulders. On genuine invariance learning without weight-tying. In *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, 2023.

Radford M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag, 1996.

E. Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1918.

Oskar Nordenfors and Axel Flinth. Ensembles provably learn equivariance through data augmentation. *arXiv preprint*, 2024.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations (ICLR)*, 2020.

Netta Ollikka, Amro Kamal Mohamed Abbas, Andrea Perin, Markku Kilpeläinen, and Stephane Deny. A comparison between humans and AI at recognizing objects in unusual poses. *Transactions on Machine Learning Research (TMLR)*, 2025.

Guillermo Ortiz-Jimenez, Apostolos Modas, Seyed-Mohsen Moosavi, and Pascal Frossard. Neural anisotropy directions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Felipe Pegado, Kimihiro Nakamura, Laurent Cohen, and Stanislas Dehaene. Breaking the symmetry: Mirror discrimination for single letters but not for pictures in the visual word form area. *NeuroImage*, 2011.

Manuel Perea, Carmen Moret-Tatay, and Victoria Panadero. Suppression of mirror generalization for reversible letters: Evidence from masked priming. *Journal of Memory and Language*, 2011.

Luis Armando Pérez Rey, Giovanni Luca Marchetti, Danica Kragic, Dmitri Jarnikov, and Mike Holenderski. Equivariant representation learning in the presence of stabilizers. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, 2023.

David Pfau, Irina Higgins, Alex Botev, and Sébastien Racanière. Disentangling by subspace diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint*, 2022.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Luca Saglietti, Stefano Mannelli, and Andrew Saxe. An analytical theory of curriculum learning in teacher-student networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Sophia Sanborn, Christian A Shewmake, Bruno Olshausen, and Christopher J. Hillar. Bispectral neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2019.

Lukas Schott, Julius Von Kügelgen, Frederik Träuble, Peter Vincent Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual representation learning does not generalize strongly within the same domain. In *International Conference on Learning Representations (ICLR)*, 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert

Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 1971.

Shoaib Ahmed Siddiqui, David Krueger, and Thomas Breuel. Investigating the nature of 3D generalization in deep neural networks. *arXiv preprint*, 2023.

Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clement Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning (ICML)*, 2021.

Mannat Singh, Laura Gustafson, Aaron Adcock, Vinicius De Freitas Reis, Bugra Gedik, Raj Prateek Kosaraju, Dhruv Mahajan, Ross Girshick, Piotr Dollar, and Laurens Van Der Maaten. Revisiting weakly supervised pre-training of visual perception models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Jascha Sohl-Dickstein, Ching Ming Wang, and Bruno A. Olshausen. An unsupervised algorithm for learning Lie group transformations. *arXiv preprint*, 2017.

Ben Sorscher, Surya Ganguli, and Haim Sompolinsky. Neural representational geometry underlies few-shot concept learning. *Proceedings of the National Academy of Sciences (PNAS)*, 2022.

Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 2002.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*, 2017.

Shobhita Sundaram, Darius Sinha, Matthew Groth, Tomotake Sasaki, and Xavier Boix. Symmetry perception by deep networks: Inadequacy of feed-forward architectures and improvements with recurrent connections. *arXiv preprint*, 2022.

Hidenori Tanaka and Daniel Kunin. Noether's learning dynamics: Role of symmetry breaking in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Sharvaree Vadgama, Mohammad Mohaiminul Islam, Domas Buracas, Christian Shewmake, Artem Moskalev, and Erik Bekkers. Probing equivariance and symmetry breaking in convolutional networks. *arXiv preprint*, 2025.

Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations (ICLR)*, 2019.

Putri A. van der Linden, Alejandro García-Castellanos, Sharvaree Vadgama, Thijs P. Kuipers, and Erik J. Bekkers. Learning symmetries via weight-sharing with doubly stochastic tensors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tycho F. A. van der Ouderaa, Alexander Immer, and Mark van der Wilk. Learning layer-wise equivariances automatically using gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Tycho F. A. van der Ouderaa, Mark van der Wilk, and Pim de Haan. Noether's razor: Learning conserved quantities. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tycho F.A. van der Ouderaa and Mark van der Wilk. Learning invariant weights in neural networks. In *Conference on Uncertainty in Artificial Intelligence*, 2022.

Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. In *International Conference on Machine Learning (ICML)*, 2022.

Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. Coordinate independent convolutional networks – isometry and gauge equivariant convolutions on riemannian manifolds. *arXiv preprint*, 2021.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Justin N. Wood. Newborn chickens generate invariant object representations at the onset of visual object experience. *Proceedings of the National Academy of Sciences (PNAS)*, 2013.

Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning (ICML)*, 2021.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite depth neural networks. In *International Conference on Learning Representations (ICLR)*, 2024a.

Jianke Yang, Nima Dehmamy, Robin Walters, and Rose Yu. Latent space symmetry discovery. In *International Conference on Machine Learning (ICML)*, 2024b.

Yongyi Yang, Core Francisco Park, Ekdeep Singh Lubana, Maya Okawa, Wei Hu, and Hidenori Tanaka. Dynamics of concept learning and compositional generalization. In *International Conference on Learning Representations (ICLR)*, 2025.

Raymond A. Yeh, Yuan-Ting Hu, Mark Hasegawa-Johnson, and Alexander Schwing. Equivariance discovery by learned parameter-sharing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow Twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*, 2021.

Richard Zemel and Geoffrey E Hinton. Discovering viewpoint-invariant relationships that characterize objects. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1990.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 2021.

Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. In *International Conference on Learning Representations (ICLR)*, 2021.

Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.