What Breaks the Curse of Dimensionality in Deep Learning?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Although learning in high dimensions is commonly believed to suffer from the curse of dimensionality, modern machine learning methods often exhibit an astonishing power to tackle a wide range of challenging real-world learning problems without using abundant amounts of data. How exactly these methods break this curse remains a fundamental open question in the theory of deep learning. While previous efforts have investigated this question by studying the data (D), model (M), and inference algorithm (I) as independent modules, in this paper we analyzes the triple (D, M, I) as an integrated system. We examine the basic symmetries of such systems, focusing on four of the main architectures in deep learning: fully-connected networks (FCN), locally-connected networks (LCN), and convolutional networks with and without pooling (GAP/VEC). By computing an eigen-decomposition of the infinite-width limits (aka Neural Kernels) of these architectures, we characterize how inductive biases (locality, weight-sharing, pooling, etc) and the breaking of spurious symmetries can affect the performance of these learning systems. Our theoretical analysis shows that for many real-world tasks it is locality rather than symmetry that provides the first-order remedy to the curse of dimensionality. Empirical results on state-of-the-art models on ImageNet corroborate our results.

1 Introduction

Statistical problems with high-dimensional data are frequently plagued by the *curse of dimensionality*, in which the number of samples required to solve the problem grows rapidly with the dimensionality of the input. Classical theory explains this phenomenon as the consequence of basic geometric and algebraic properties of high-dimensional spaces; for example, the number of ϵ-cubes inside a unit cube in $\mathbb{R}^d$ grows exponentially like ϵ^{-d}, and the number of degree r polynomials in $\mathbb{R}^d$ grows like a power-law d^r. Since for real-world problems d is typically in the hundreds or thousands, classical wisdom suggests that learning is likely to be infeasible. However, starting from the groundbreaking work AlexNet [1], practitioners in deep learning have tackled a wide range of difficult real-world learning problems ([2–6]) in high dimensions, once believed by many to be out-of-scope of current techniques. The astonishing success of modern machine learning methods clearly contradicts the curse of dimensinonality and therefore poses the fundamental question: mathematically, how do modern machine learning methods break the curse of dimensionality?

To answer this question, we must trace back to the most fundamental ingredients of machine learning methods. They are the data ($\mathcal{D}$), the model ($\mathcal{M}$), and the inference algorithm ($\mathcal{I}$).

Data ($\mathcal{D}$) is of course central in machine learning. In the classical learning theory setting, the learning objective usually has a power-law decay $m^{-\beta}$ as the function of the number of training samples m. The theoretical bound on β is usually tiny, owing to the curse of dimensionality, and is of

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

limited practical utility for high-dimensional data. On the other hand, empirical measurements of β in state-of-the-art deep learning models typically reveal values of β that are not at all small (e.g. $\beta = 0.43$ for ResNet in Fig.4) even though d is quite large (e.g. $d \sim 10^5$ for ImageNet). This example suggests that the learning curve must have important functional dependence on $\mathcal{M}$ and $\mathcal{I}$. Indeed, as we will observe later, many of the best performing methods exhibit learning curves for which $\beta = \beta(m)$ actually *increases* as m becomes larger, i.e. data makes the usage of data more efficient. We call this phenomenon DIDE, for **d**ata **i**mproves **d**ata **e**fficiency.

Designing machine learning models ($\mathcal{M}$) that maximize data-efficiency is critical to the success of solving real-world tasks. Indeed, breakthroughs in machine learning are often driven by novel architectures LeNet [7], AlexNet[1], Transformer [2], etc. While some of the inductive biases of these methods are clear (e.g. translation symmetries of CNNs), others tend to build off of prior empirical success and are less well-understood (e.g. the implicit bias of SGD). To build our understanding of these biases and how they affect learning, we conduct a theoretical analysis of them in the infinite-width setting [8–12], which preserves most salient aspects of the architecture while enabling tractable calculations. We classify all phenomena that could be explained by infinite networks alone as the consequences of inductive biases.

The inference procedure ($\mathcal{I}$) is what enables *learning* in machine learning methods. It is widely believed that modern inference methods, specifically gradient descent and variants, 'implicitly' bias the solutions of the networks towards those that generalize well and away from those that generalize poorly [13–15]. The effects of the inference algorithm are intimately tied to the specifics of the model (e.g. weight-sharing) and the data (e.g. augmentation), and might not be fully understood with a fixed-data, fixed-model analysis. Indeed, good performance may derive from interactions between $(\mathcal{M}, \mathcal{I})$, or $(\mathcal{D}, \mathcal{I})$, or even $(\mathcal{D}, \mathcal{M}, \mathcal{I})$. In Sec. 3.1, we demonstrate the DIDE effect for a particular choice of $(\mathcal{D}, \mathcal{M}, \mathcal{I})$ and show that this effect disappears if any one of $\mathcal{D}$, $\mathcal{M}$, or $\mathcal{I}$ is altered.

The above discussion highlights the insufficiency of treating $\mathcal{D}$, $\mathcal{M}$, and $\mathcal{I}$ as separate non-interacting modules. They must be considered as an integrated system. Throughout this paper, we will refer to the triplet $(\mathcal{D}, \mathcal{M}, \mathcal{I})$ as a (machine) learning system and the tuple $(\mathcal{M}, \mathcal{I})$ as the learning algorithm of the system that operates on $\mathcal{D}$. We summarize our contributions below.

1. We surface the basic symmetries of various $(\mathcal{D}, \mathcal{M}, \mathcal{I})$associated to four of the main architectures in deep learning FCN_n (fully-connected networks), LCN_n (locally-connected networks), $\mathsf{VEC}_n/\mathsf{GAP}_n$ (convolution networks with a flattening /a global average pooling readout layer), their infinite width counterparts $\mathsf{FCN}_\infty/\mathsf{LCN}_\infty/\mathsf{VEC}_\infty/\mathsf{GAP}_\infty$. Treating $\mathsf{FCN}_{n/\infty}$ as the baseline model, we show that the locality from LCN_n and the weight-sharing from $\mathsf{VEC}_n/\mathsf{GAP}_n$ break spurious symmetries and lead to better systems. Empirically, we examine the relation between the symmetries and the performance of the systems in the infinite width setting and finite width setting with various of interventions. Surprisingly, we observe that state-of-the-art learning system (EfficientNet[16]) on ImageNet can learn almost equally well even the coordinate of the data are transformed by the symmetry group defined by LCN_n.
2. We show that although the weight-sharing from VEC_n provides coordinate information of the data to the system, as the width gets larger, it becomes harder for the learning algorithm to explore such information and at infinite width, the system restores the symmetry group that is identical to LCN_n, and is completely unaware of the coordinate information. As a consequence, the performance of the network, as a function of width, monotonically decays [12]. This is in stark contrast to recent finding that the performance of network is positively correlated to its width. We show that this phenomenon continues to hold even with various interventions (larger learning rate and l2 regularization) to the training procedures. However, with more data (e.g. data augmentation) VEC_n can be on par with GAP_n.
3. The function space defined by LCN_n is a super set of that defined by VEC_n. We prove the opposite is true. Therefore, VEC_n is able to express functions in the space with a stronger inductive bias GAP_n (translation invariance) and functions in a seemingly much larger class LCN_n. We hypothesize that as the dataset grows, the learned functions using VEC_n is transitioned away from those learned using LCN_n and become closer to those learned using GAP_n. This suggests, even though the prior (provided by human) is not 100% correct, with the help of more data, gradient descent might be able to correct it, a possible explanation of DIDE.

4. When the input space is the product of hyperspheres, we eigendecompose the kernels associated to one-hidden layer infinite width network, FCN_∞, $\mathsf{VEC}_\infty = \mathsf{LCN}_\infty$ and GAP_∞. We treat FCN_∞ as the baseline, whose order r eigenspace has dimension of order d^r and eigenvalues of order d^{-r} for $r \geq 0$ [17]. We show that locality alone (i.e. VEC_∞) dramatically reduces the dimension of the r-eigenspace for $r \geq 2$ and the spectral gap between all r-eigenspaces but $r = 0$ and $r = 1$, making learning of higher order eigenspaces feasible with dramatically fewer samples and gradient steps. In addition, pooling (i.e. GAP_∞) reduces the dimension of r-eigenspace for $r \geq 1$ by a factor equal to the size of the pooling window, but it does not change the spectra in an essential way.

Our empirical and theoretical results surface the importance of locality which, we believe, provides the first-order remedy to the curse of dimensionality for many real-world tasks and which has been largely overlooked.

2 Preliminary and Notation

2.1 Neural Networks

We focus our presentation on the supervised learning setting and more concretely, on image recognition. Let $\mathcal{D} \subseteq (\mathbb{R}^d)^3 \times \mathbb{R}^k \equiv \mathbb{R}^{3d} \times \mathbb{R}^k$ denote the data set (training and test) and $\mathcal{X} = \{x : (x, y) \in \mathcal{D}\}$ and $\mathcal{Y} = \{y : (x, y) \in \mathcal{D}\}$ denote the input space (images) and label space, respectively. Here d is the spatial dimension (e.g. $d = 32 \times 32$ for CIFAR-10) of the images and 3 is the total number of channels (i.e. RGB). We use FCN_n to denote a L-hidden layer fully-connected network with identical hidden widths $n_l = n \in \mathbb{N}$ for $l = 1, ..., L$ and with readout width $n_{L+1} = k$ (the number of logits). For each $x \in \mathbb{R}^{3d} = (\mathbb{R}^d)^3$, we use $h^l(x), x^l(x) \in \mathbb{R}^{n_l}$ to represent the pre- and post-activation functions at layer l with input x. The recurrence relation FCN_n is given by

$$\begin{cases} h^{l+1} &= x^l W^{l+1} \\ x^{l+1} &= \phi\left(h^{l+1}\right) \end{cases} \text{ and } W^l_{i,j} = \frac{1}{\sqrt{n_l}} \omega^l_{ij}, \quad \omega^l_{ij} \sim \mathcal{N}(0, 1) \tag{1}$$

where ϕ is a point-wise activation function, $W^{l+1} \in \mathbb{R}^{n_l \times n_{l+1}}$ are the weights and ω^l_{ij} are the trainable parameters, drawn i.i.d. from a standard Gaussian $\sim \mathcal{N}(0, 1)$ at initialization. For simplicity of the presentation, the bias terms and the hyperparameters (the variances of the weights) are omitted. Adding them back won't affect the conclusion of the paper.

For convolutional networks or locally-connected networks, the inputs are treated as tensors in $(\mathbb{R}^d)^3$. The recurrent relation of convolutional networks can be written as

$$x^{l+1}_{\alpha,j} = \phi(h^{l+1}_{\alpha,j}) \quad \text{and} \quad h^{l+1}_{\alpha,j} \equiv \frac{1}{\sqrt{(2k+1)n^l}} \sum_{j=1}^{n^l} \sum_{\beta=-k}^{k} x^l_{\alpha+\beta,i} \omega^l_{ij,\beta} \tag{2}$$

Here $\alpha \in [d]$ denote the spatial location, $i/j \in [n]$ denotes the fanin/fanout channel indices. For notational convenience, we assume circular padding and stride equal to 1 for all layers. The features of the penultimate layer are 2D tensors and there are two commonly used approaches to map them to the logit layer: stack a dense layer after either vectorizing the 2D tensor to a 1D vector or applying a global average pooling layer to each channel. We use $\mathsf{VEC}_n/\mathsf{GAP}_n$ to denote the network obtain from the former/latter, which are known to be equipped with the inductive biases translation equivariant/invariant. The readout layer of $\mathsf{VEC}_n/\mathsf{GAP}_n$ could be written as

$$x^{L+1}_j = \frac{1}{\sqrt{dn}} \sum_{\alpha \in [d]} x^L_{\alpha,i} w^{L+1}_{\alpha,ij}\ , \quad x^{L+1}_j = \frac{1}{\sqrt{n}} \sum_{i \in [n]} \left(\frac{1}{d} \sum_{\alpha \in [d]} x^L_{\alpha,i} \right) w^{L+1}_{ij} \tag{3}$$

We briefly remark the the key difference between the two. In VEC_n, each pixel in the penultimate layer has its own (independent random) variable while pixels within the same channel shared the same (random) variable in GAP_n. It is clear that the function space of VEC_n contains that of GAP_n. Locally Connected Networks LCN_n [18, 19] are convolutional network *without* weight sharing between spatial locations. LCN_n preserve the connectivity pattern, and thus topology, of a convnet. Mathematically, the current formula is defind as in Equation 2 with all the *shared* parameters $\omega^l_{ij,\beta}$ replaced by *unshared* $\omega^l_{ij,\alpha,\beta} \sim \mathcal{N}(0, 1)$

In this note, we assume that the LCN_n are always associated with a vectorization readout layer and it is clear, as a function space, LCN_n is a super set of VEC_n. Interestingly,the opposite is also true.

Theorem 2.1 (Sec. B)**.** *Let* $\mathsf{VEC}_n/\mathsf{LCN}_n/\mathsf{GAP}_n$ *denote the set of functions that can be represented by L-hidden layer* $\mathsf{VEC}_n/\mathsf{LCN}_n/\mathsf{GAP}_n$ *networks with hidden width* n*. Then*

$$\mathsf{GAP}_n \subseteq \mathsf{VEC}_n \subseteq \mathsf{LCN}_n \subseteq \mathsf{VEC}_{dn} \tag{4}$$

The significance of this theorem is that if we consider the function space VEC_n as a soft *prior*, gradient descent could move it closer to a *better* prior GAP_n (translation invariance) if the average pooling is (approximately) learned in the readout layer or it might remain close to LCN_n.

2.2 Gradient Descent Training

We use f to denote any functions defined by the architectures above and θ to denote the collection of all parameters. Denote by θ_t the time-dependence of the parameters and by θ_0 their initial values. We use $f_t(x) \equiv f(x, \theta_t) \in \mathbb{R}^k$ to denote the output (or logits) of the neural network at time t. Let $\ell(\hat{y}, y) : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ denote the loss function where the first/second argument is the prediction/true label. By applying continuous time gradient descent to minimize the objective $\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \ell(f_t(x, \theta), y)$, the evolution of the parameters θ and the logits f can be written as

$$\dot{\theta}_t = -\nabla_\theta f_t(\mathcal{X}_T)^T \nabla_{f_t(\mathcal{X}_T)}\mathcal{L}\,, \qquad \dot{f}_t(\mathcal{X}_T) = \nabla_\theta f_t(\mathcal{X}_T)\,\dot{\theta}_t = -\,\hat{\Theta}_t(\mathcal{X}_T, \mathcal{X}_T)\nabla_{f_t(\mathcal{X}_T)}\mathcal{L} \tag{5}$$

where $f_t(\mathcal{X}_T) = \mathrm{vec}\left([f_t\,(x)]_{x\in\mathcal{X}_T}\right)$, the $k|\mathcal{D}| \times 1$ vector of concatenated logits for all examples, and $\nabla_{f_t(\mathcal{X}_T)}\mathcal{L}$ is the gradient of the loss with respect to the model's output, $f_t(\mathcal{X}_T)$. $\hat{\Theta}_t \equiv \hat{\Theta}_t(\mathcal{X}_T, \mathcal{X}_T)$ is the tangent kernel at time t, which is a $k|\mathcal{D}| \times k|\mathcal{D}|$ kernel matrix

$$\hat{\Theta}_t = \nabla_\theta f_t(\mathcal{X}_T)\nabla_\theta f_t(\mathcal{X}_T)^T \tag{6}$$

One can define the tangent kernel for general arguments, e.g. $\hat{\Theta}_t(x, \mathcal{X}_T)$ where x is test input. At finite-width, $\hat{\Theta}$ will depend on the specific random draw of the parameters and evolve with time. As such, for a test point x the prediction $f_t(x)$ depends on the random initalization and is also stochastic. Note that the parameters are initialized randomly and the randomness will be carried out through the training procedure. As a consequence, the prediction functions are stochastic.

2.3 Infinite Network: Gaussian Processes and the Neural Tangent Kernels

Neural Networks as Gaussian Processes (NNGP). As the width $n \to \infty$, at initialization the output $f_0(\mathcal{X})$ forms a Gaussian Process $f_0(\mathcal{X}) \sim \mathcal{GP}(0, \mathcal{K}(\mathcal{X}, \mathcal{X}))$, known as the NNGP [8, 20, 21]. Here $\mathcal{K}$ is the GP kernel and can be computed in closed form for a variety of architectures. By treating this infinite width network as a Bayesian model (aka Bayesian Neural Networks) and applying Bayesian inference, the posterior is also a GP

$$\mathcal{N}\left(\mathcal{K}(\mathcal{X}_*, \mathcal{X}_T)\mathcal{K}^{-1}(\mathcal{X}_T, \mathcal{X}_T)\mathcal{Y}_T, \mathcal{K}(\mathcal{X}_*, \mathcal{X}_*) - \mathcal{K}(\mathcal{X}_*, \mathcal{X})\mathcal{K}(\mathcal{X}, \mathcal{X})^{-1}\mathcal{K}(\mathcal{X}_*, \mathcal{X})^T\right) \tag{7}$$

Neural Tangent Kernelss(NTK). Recent advance in global convergence theory of over-parameterized networks [22–25, 12] has shown that under certain assumptions, the tangent kernels is almost stationary over the course of training and is concentrated on its infinite width limit Θ in the sense there is a constant C independent of t and the network's width n such that

$$\sup_{t\geq 0} \|\hat{\Theta}_t(\mathcal{X}_T, \mathcal{X}_T) - \Theta(\mathcal{X}_T, \mathcal{X}_T)\|_F + \|\hat{\Theta}_t(\mathcal{X}_T, \mathcal{X}_*) - \Theta(\mathcal{X}_T, \mathcal{X}_*)\|_F \leq \frac{C}{\sqrt{n}}. \tag{8}$$

where is the infinite width limit of Θ at initialization, whose existence has been proved in [22, 26]. As such, when the loss is the mean squared error (MSE), the mean prediction (marginarized over random initialization) has the following closed form

$$f(\mathcal{X}_*) = \Theta\left(\mathcal{X}_*, \mathcal{X}_T\right)\Theta^{-1}(\mathcal{X}_T, \mathcal{X}_T)\left(I - e^{-\eta\Theta(\mathcal{X}_T, \mathcal{X}_T)t}\right)\mathcal{Y}\,, \tag{9}$$

Letting $t \to \infty$, the above solution is the same as that of the kernel ridgeless regression using the infinite width tangent kernel Θ. We use $\mathsf{FCN}_\infty(x)$, $\mathsf{LCN}_\infty(x)$, $\mathsf{VEC}_\infty(x)$ and $\mathsf{GAP}_\infty(x)$ to denote the infinite width solutions (either the GP inference or the NTK regression) for the corresponding architectures, where we have suppressed the dependence on the training data $(\mathcal{X}_T, \mathcal{Y}_T)$.

3 Symmetries of Machine Learning Systems

Symmetry is fundamental in physical systems. So is it in machine learning systems. We explore symmetries of various machine learning systems in this section. Given $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ and a transformation on the input space $\tau : \mathbb{R}^{3d} \to \mathbb{R}^{3d}$, we set $\tau(\mathcal{D}) = (\tau(\mathcal{X}), \mathcal{Y})$. Let $\mathrm{O}(3d)$ denote the orthogonal group on the flatten input space $\mathbb{R}^{3d}$. The subgroup $\mathrm{O}(3)^d \leq \mathrm{O}(3d)$ operates on the un-flattened input $(\mathbb{R}^d)^3$, whose element rotates each pixel $x_\alpha \in \mathbb{R}^3$ by an independent element $\tau_\alpha \in \mathrm{O}(3)$. The smaller subgroup $\mathrm{O}(3) \otimes \mathbf{I}_d \leq \mathrm{O}(3)^d$ applies the *shared* rotation (i.e. $\tau_\alpha = \tau$ to all x_α for $\alpha \in [d]$). We use $\mathrm{P}(3d)$ to denote the permutation group on $\mathbb{R}^{3d}$ and $\mathrm{P}(3)^d$ and $\mathrm{P}(3) \otimes \mathbf{I}_d$ are defined similarly. Note that rotating $\mathcal{X}$ by τ is equivalent to transfer the original coordinate system by the adjoint tranformation $\tau^* = \tau^{-1}$.

For a deterministic (stochastic) learning algorithm $\mathcal{A} = (\mathcal{M}, \mathcal{I})$, we use $\mathcal{A}(\mathcal{D}_T)$ to denote the learned function (distribution of the learned functions) using training set $\mathcal{D}_T$. We use $\mathcal{A}^\tau(\mathcal{D}_T)$ to denote the learned function(s) using $\tau(\mathcal{D}_T)$ and makes prediction on the transformed test point $\tau(\mathcal{X}_*)$. In another word, the learning algorithm is conducted in the input space whose coordinate system is transformed by τ^{-1}.

Definition 1. *Let $\mathcal{G}$ be a group of transformations $\mathbb{R}^{3d} \to \mathbb{R}^{3d}$. We say a deterministic (stochastic) learning algorithm $\mathcal{A} = (\mathcal{M}, \mathcal{I})$ is g-invariant if $\mathcal{A} = \mathcal{A}^g$ ($\mathcal{A} =^d \mathcal{A}^g$). In this case, we say the system $(\mathcal{D}, \mathcal{M}, \mathcal{I})$ is g-invariant and use the notation $(\mathcal{D}, \mathcal{M}, \mathcal{I}) = (g\mathcal{D}, \mathcal{M}, \mathcal{I})$. If this holds for all $g \in \mathcal{G}$, then we say the algorithm and the system are $\mathcal{G}$-invariant.*

If $(\mathcal{M}, \mathcal{I})$ is the algorithm of minimum norm linear regressor, then $(\mathcal{D}, \mathcal{M}, \mathcal{I})$ is $\mathrm{O}(3)^d$-invariant; see Sec.F for more details. Note that the symmetry (invariance) in our definition is a property of a system and is different from the notion of symmetry that are commonly used in the machine learning community, which is a property of a function (e.g. translation invariance).

Theorem 3.1 (Sec.C). *If the parameters of the networks are initialized with iid $\mathcal{N}(0, 1)$, then*

- $\mathsf{FCN}_{n/\infty}$ are $\mathrm{O}(3d)$-invariant.
- $\mathsf{LCN}_{n/\infty}$ are $\mathrm{O}(3)^d$-invariant.
- VEC_n is $\mathrm{O}(3) \otimes \mathbf{I}_d$-invariant and VEC_∞ is $\mathrm{O}(3)^d$-invariant.
- $\mathsf{GAP}_{n/\infty}$ are $\mathrm{O}(3) \otimes \mathbf{I}_d$-invariant.

The $\mathrm{O}(3d)$-invariant of FCN_∞ is because the NTK/NNGP kernel is an inner product kernel, namely, there is a function k such that the kernels have the form $k(\langle x, x' \rangle)$. The $\mathrm{O}(3d)$-invariant of finite width FCN_n is due to the Gaussian initialization of the first layer which was first observed and proved in [27]. Rotating the input by $\tau \in \mathrm{O}(3d)$ is equivalent to rotating the weight matrix ω of the first layer by τ^*. Since for $\omega \in \mathcal{N}(0, 1)^{3d}$ $\tau^* \omega =^d \omega$, at random initialization, the distribution of the output functions (the prior) are unchanged if all inputs are rotated by the same element in $\mathrm{O}(3d)$. This property continues to hold throughout the course of (continue/discrete) gradient descent training with/without L^2-regularization and Bayesian posterior inference. For the same reason, LCN_n is $\mathrm{O}(3)^d$-invariant because each patch of the image uses independent Gaussian random variables. However, weight-sharing in VEC_n and GAP_n breaks the $\mathrm{O}(3)^d$ symmetry, reducing it to $\mathrm{O}(3) \otimes \mathbf{I}_d$.

For infinite networks, $\mathsf{LCN}_\infty = \mathsf{VEC}_\infty$ [28–31]. The kernels of VEC_∞ and GAP_∞ are of the forms

$$\Theta_{\mathsf{VEC}}(x, x') = k(\{\langle x_\alpha, x'_\alpha \rangle\}_{\alpha \in [d]}) \quad \text{and} \quad \Theta_{\mathsf{GAP}}(x, x') = k(\{\langle x_\alpha, x'_{\alpha'} \rangle\}_{\alpha, \alpha' \in [d]}), \tag{10}$$

resp. The former depends only on the inner product between pixels in the *same* spatial location, breaking the $\mathrm{O}(3d)$ symmetry and reducing it to $\mathrm{O}(3)^d$. In addition, the latter depends also on the inner products of pixels across different spatial locations due to pooling, which breaks the $\mathrm{O}(3)^d$ symmetry and reduces it to $\mathrm{O}(3) \otimes \mathbf{I}_d$.

Note that $\dim(\mathrm{O}(3d)) = 3d(3d-1)/2$, $\dim(\mathrm{O}(3)^d) = 3d$ and $\dim(\mathrm{O}(3) \otimes \mathbf{I}_d) = 3$. $\mathsf{LCN}_n/\mathsf{VEC}_\infty$ dramatically reduces the dimension of the symmetry group. It is worth mentioning that while $\dim(\mathrm{O}(3d))$ many pairs of rotated and unrotated images are needed to recover the exact rotation in $\mathrm{O}(3d)$, only 3 pairs are sufficient for $\mathrm{O}(3)^d$, same as that of $\mathrm{O}(3) \otimes \mathbf{I}_d$. The results of the paper are presented in the most *vanilla* setting. Our methods can easily extend to more complicated architectures like ResNet[32], MLP-Mixer[33] and etc. The symmetry groups of such systems need to be computed in a case-by-case manner by identifying the invariant group of the random initialization and training procedures.

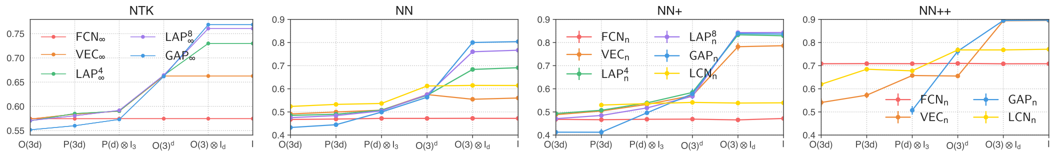

Figure 1: **Performance vs Symmetry**. Machine learning systems are equipped with various kinds of symmetries. Transforming the system by the associated symmetry does not affect the performance of the system. However, injecting spurious symmetries beyond the associated symmetries could dramatically degrade their performance for both finite and infinite networks.

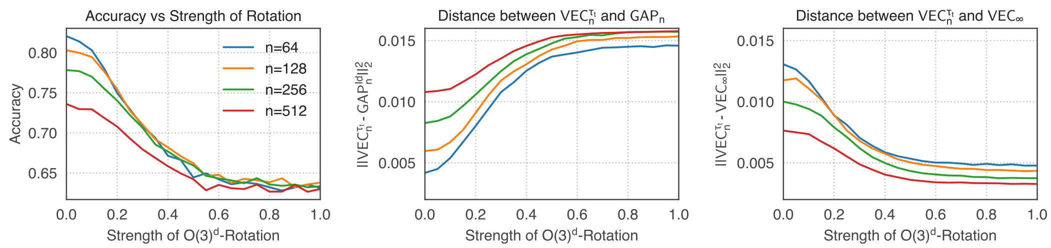

Figure 2: Even in the NN+ setting, VEC_n is closer to GAP_n for small n and moves towards VEC_∞ with more symmetries and/or larger n and accuracy drops.

3.1 Empirical Supports and Observations

Performance under Rotations. We examine the performance of: FCN, VEC, LCN, GAP and $\mathsf{LAP}^{4/8}$, when the coordinates of the data are transformed by six different groups (x-axis in Fig.1) using the standard dataset CIFAR-10. , Here $\mathsf{LAP}^{4/8}$ is the same as GAP except the readout layer is replaced by the **L**ocal **A**verage **P**ooling with window size $4\times 4/8\times 8$. We consider 4 types of training methods: (1) NTK, i.e. infinite networks (2)NN, our baseline for finite width neural network which is trained with momemtum using a small learning rate and without L^2 regularizer and the network is centered (+C) to reduce the variance from random initialization (3)NN+= NN+LR+L2−C, i.e. using a larger learning rate (+LR), adding L^2 regularization (+L2)) and removing the centering (−C) (4) NN++=NN++DA, adding MixUp[34] data augmentation (+DA) to NN+. Overall, we observe that, for most of the cases in NTK/NN/NN+, adding spurious symmetry to a system $(\mathcal{D}, \mathcal{M}, \mathcal{I})$ degrades the performance towards that of the system invariant to that symmetry. Surprisingly, in the baseline NN, performance of $\mathsf{VEC}_n + \mathrm{O}(3)\otimes\mathbf{I}_d$ rotation is slightly worse than that of $\mathsf{VEC}_n + \mathrm{O}(3)^d$ and than that of LCN_n, indicating that the system with $\mathcal{M} = \mathsf{VEC}_n$ is likely operating closely on the $\mathrm{O}(3)^d$ symmetry. The interventions −C+L2+LR in NN+ distinguishes the performance of $\mathsf{VEC}_n + \mathrm{O}(3)\otimes\mathbf{I}_d$ from $\mathsf{VEC}_n + \mathrm{O}(3)^d$ and +DA eventually closes the performance gap between $\mathsf{VEC}_n + \mathrm{O}(3)\otimes\mathbf{I}_d$ and $\mathsf{GAP}_n + \mathrm{O}(3)\otimes\mathbf{I}_d$, helping the system to be aware of the smaller symmetry $\mathrm{O}(3)\otimes\mathbf{I}_d$ and escaping from the $\mathrm{O}(3)^d$ symmetry.

Symmetry Breaking of VEC_n. Assuming Equation 8, namely, the network is in the NTK regime,

$$\lim_{n\to\infty} |\mathbb{E}\mathsf{VEC}_n(x) - \mathsf{VEC}_\infty(x)| + \lim_{n\to\infty} |\mathbb{E}\mathsf{VEC}_n(x) - \mathbb{E}\mathsf{VEC}_n^\tau(x)| \leq Cn^{-\frac{1}{2}} \tag{11}$$

where the expectation $\mathbb{E}$ is over random initialization and $\mathsf{VEC}_n(x)$ is the prediction of the test point x when $t=\infty$, i.e. training loss is 0. VEC_n^τ is the prediction of the τ-rotated system, $\tau \in \mathrm{O}(3)^d$. The $\mathrm{O}(3)^d$ symmetry is restored as $n \to \infty$. As such, for large n, the system is approximately $\mathrm{O}(3)^d$-invariant. In Figs. 2, we randomly sample a $\tau \in \mathrm{O}(3)^d$ and use the exponential map to construct a continuous interpolation $\tau_t \in \mathrm{O}(3)^d$ between $\tau_0 = \mathbf{Id}$ and $\tau_1 = \tau$. We train the network as in NN+ (+LR+L2−C) using different n and τ_t and average the predictions over 10 random initialization as an approximation of $\mathbb{E}\mathsf{VEC}_n^{\tau_t}(x)$. Not surprisingly, as n increases and/or t increases, (1) test performance decays monotonically (left panel in Fig.2), (2) the distance to $\mathbb{E}\mathsf{GAP}_n$ increases monotonically (middle panel) and (3) distance to VEC_∞ decrease monotonically (right panel). Clearly, the coordinate information from the data is utilized by smaller width VEC_n.

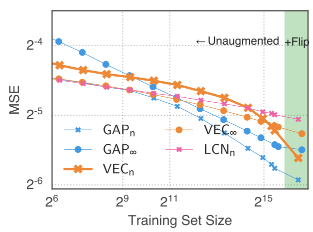

Figure 3: **Data Bends Learning Curve of** VEC_n**.** We study the effect of training set size to the network's performance for various models. In the small dataset regime, the slope of the learning curve (in the log-log plot) of VEC_n is similar to that of VEC_∞ and FCN_n. However, as the dataset gets larger, the slope increases significantly. This is hinted by Theorem 2.1.

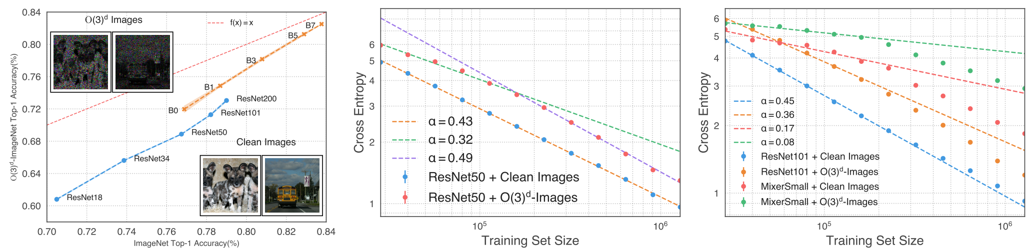

Figure 4: With coordinate of the input data rotated by $\mathrm{O}(3)^d$, state of the art models learn as good as without rotation. middle/right: slopes of the learning curves increases due to more data. DIDE

DIDE for VEC_n**.** To understand the role of data, we vary the training set size of Cifar10 from about 2^6 to 50k (the whole un-augmented training set) and to 100k (adding left-right flip augmentation) and plot the learning curves in Fig.3. We observe dramatic speedup of learning for VEC_n in the larger data set regime, which isn't the case for VEC_∞ (kernel), LCN_n, GAP_∞ and even for GAP_n after $m = 2^{12}$. We argue that this is due to the prior (the function space defined by the model) is too large (and not optimal) for the task and the coupled effect of more data together with inference procedures corrects the prior, as it is suggested by Theorem 2.1.

DIDE for SOTA models. In the middle and right panels of Fig.4, we provide additional evidence in a larger scale setting. We generate learning curves of ImageNet using ResNet50 and (MLP-)Mixer [33], a very recent architecture that contains no convolution layers except the first layer, which is a convolution with filter size and stride equal to $(16, 16)$ (patches are disjoint). The symmetry group associated to (first layer of) ResNet ($\mathrm{O}(2^2 \times 3) \otimes \mathbf{Id}_{112\times 112}$) is similar to that of GAP_n which is relatively small. However, the symmetry group induced by the first layer of the Mixer is $\mathrm{O}(3 \times 16^2) \otimes \mathbf{I}_{14^2}$, where 3×16^2 is number of entries in the $(16, 16, 3)$ patch (RGB channels) and $14^2 = 224^2/16^2$ is the number of patches. Although the dimension of $\mathrm{O}(3 \times 16^2) \otimes \mathbf{I}_{14^2}$ is quite large (about $(3 \times 16^2)^2/2$), it is still dramatically smaller than that of applying a fully-connected layer to the flatten images, which $\mathrm{O}(3 \times 224^2)$ (whose dimension is $(3 \times 224^2)^2/2$). In the middle panel of Fig.4, we observe an almost perfect power-law scaling for the learning curve for the ResNet50 system with unrotated images. When the images are rotated by $\mathrm{O}(3)^d$ ($d = 224^2$), the learning curve is relatively flat in the smaller data regime (green dashed line). However, as the data set grows, it eventually catches up (purple dashed line) as that of the unrotated setting; see Sec.E for ResNet34/101. In the third panel, we see the learning curves are much flatter (red) for the Mixer and even more so for the rotated images (green). Again, these curves are bent towards that of ResNet50 with unrorated images as data increases, indicating the prior was being corrected.

Finally, in the left panel of Fig.4, we compare the accuracy of state-of-the-art models trained on both unrotated and $\mathrm{O}(3)^d$ rotated images. Surprisingly, the gap between the two are not large and becomes smaller for better performant models. For EfficientNet B7 [1], the top-1 accuracy of the rotated system is only 1.2% off from the unrotated one. See Fig.S7, S8 and S9 for rotated and unrotated images.

4 Eigenecomposition of Neural Kernels

To gain insight into the inductive biases of various architectures, we eigendecompose the kernels using spherical harmonics. We assume the input space $\mathcal{X} = \{\xi = (\xi_0, \ldots, \xi_{p-1}) \in (\sqrt{d_0}\mathbb{S}^{(d_0-1)})^p\} \subseteq$

[1]Still under training

$\mathbb{R}^{d_0 p}$, i.e. the p-product of (d_0-1)-sphere with radius $\sqrt{d_0}$. We call $\xi_i \in \sqrt{d_0}\mathbb{S}^{(d_0-1)}$ a mini-patch and $(\xi_i, \xi_{i+1}, \dots, \xi_{i+s-1}) \in (\sqrt{d_0}\mathbb{S}^{(d_0-1)})^s\}$ a patch for $i \in [p]$, where circular boundary condition is assumed. We consider the asymptotic limit when $d_0 = d^\alpha, p = d^{1-\alpha}$ and $d = pd_0 \to \infty$ and treat $0 < \alpha < 1$ and s as fixed constant. The input space $\mathcal{X}$ is associated with the product measure $\mu \equiv \sigma_{d_0}^p$, where σ_{d_0} is the normalized uniform measure on $\sqrt{d_0}\mathbb{S}^{(d_0-1)}$. The kernels associated to the one-hidden layer infinite networks (NNGP and NTK) have the following general forms

$$k\left(\frac{1}{p}\sum_{i\in[p]} \xi_i^T \eta_i / d_0\right), \;\; \frac{1}{p}\sum_{i\in[p]} k\left(\frac{1}{s}\sum_{b\in[s]} \xi_{i+b}^T \eta_{i+b}/d_0\right), \;\; \frac{1}{p^2}\sum_{i,j\in[p]} k\left(\frac{1}{s}\sum_{b\in[s]} \xi_{i+b}^T \eta_{j+b}/d_0\right), \tag{12}$$

for $\mathcal{K}_{\mathsf{FCN}}, \mathcal{K}_{\mathsf{VEC}}$ and $\mathcal{K}_{\mathsf{GAP}}$, resp. Note that the exact form of the (positive definite) kernel function $k : \mathbb{R} \to \mathbb{R}$ depends on the type of the kernels (NNGP vs NTK), activations, hyperparameters and etc. We assume the kernel is sufficiently smooth in $(-1, 1)$ and the Tayor expansion of $k^{(r)}$ converges uniformly in $[-1, 1]$ for sufficiently many $r \in \mathbb{N}$. We use the notation that $A \sim B$ if there are positive constants c and C independent of d such that $cA \le B \le CA$ for d sufficiently large. We use $\mathcal{K}$ to represent any kernels above and consider it as a Hilbert–Schmidt operator on $L^2(\mathcal{X}, \mu)$

$$\mathcal{K}f(\xi) = \int_{\mathcal{X}} \mathcal{K}(\xi, \eta) f(\eta) d\mu\,, \quad f \in L^2(\mathcal{X}, \mu), \tag{13}$$

which is well-defined since μ is a probability measure and k is bounded. Let $\vec{r} = (r_0, \dots, r_{p-1}) \in \mathbb{N}^p$, τ the shifting operator $\tau\vec{r} = (r_{p-1}, r_0, \dots, r_{p-2})$. The s-banded subset of $\mathbb{N}^p$ is defined to be

$$B(\mathbb{N}^p, s) = \{\vec{r} \in \mathbb{N}^p : \mathrm{dist}(\mathrm{argmax}_j r_j \neq 0, \mathrm{argmin}_j r_j \neq 0) \le s - 1\} \tag{14}$$

which is a quantifier used to restrict the support of a function on a patch. Here $\mathrm{dist}(i, j) = \min\{|i - j|, p - |i - j|\}$, a distance defined on the cyclic group $[p] = \mathbb{Z}/p\mathbb{Z}$. The quotient space $B(\mathbb{N}^p, s)/\tau$ denotes a subset of $B(\mathbb{N}^p, s)$ by identifying $\vec{v} = \vec{v}'$ as the same element if $\vec{v} = \tau^a \vec{r}'$ for some $a \in [p]$. Finally, $Y_{r_j, l_j}(\xi_j)$ is used to denote the l_j-th spherical harmonic of degree r_j in the unit sphere $\mathbb{S}^{(d_0-1)}$ and has unit norm under the normalized measure on $\mathbb{S}^{(d_0-1)}$. As such $Y_{r_j,l_j}(\xi_j/\sqrt{d_0}) \in L^2(\sqrt{d_0}\mathbb{S}^{(d_0-1)}, \sigma_{d_0})$ has unit norm. Recall that the total number of spherical harmonic of degree r_j in $\mathbb{S}^{(d_0-1)}$ is $N(d_0, r_j) = (2r_j + d_0 - 2)\binom{d_0+r_j-3}{r_j-1}/r_j \sim d_0^{r_j}/r_j!$ as $d_0 \to \infty$. We use $N(d_0, \vec{r}) = \prod_{j\in[p]} N(d_0, r_j)$ and $[N(d_0, \vec{r})] = \prod_{j\in[p]}[N(d_0, r_j)]$, resp. Let

$$Y_{\vec{r},\vec{l}}(\xi) = \prod_{j\in[p]} Y_{r_j,l_j}(\xi_j) \tag{15}$$

The following theorem shows that locality (VEC_∞) dramatically reduces both the dimensions of $r \ge 1$ eigenspaces and the spectral gap between them. In addition, pooling (i.e. translation symmetry of GAP_n) reduces their dimensions by an additional factor of p. See Sec.E for the implication of this theorem to learning.

Theorem 4.1. *[Sec.D] We have the following eigendecomposition for the integral operator $\mathcal{K}$*

$$\mathsf{H} = \bigcup_{r\in\mathbb{N}} \mathsf{H}^{(r)} = \bigcup_{r\in\mathbb{N}} \bigcup_{\vec{r}\in Q(\mathcal{K},r)} \mathsf{H}^{(\vec{r})}, \tag{16}$$

where $Q(\mathcal{K}, r)$ is a quantifier defined below. If $r = 0$, $\mathsf{H}^{(0)}$ is the space of constant functions and the eigenvalue is $\sim k(0)$. For $r \ge 1$, we have the following.

(1)Baseline: $\mathcal{K} = \mathcal{K}_{\mathsf{FCN}}$. $Q(\mathcal{K}, r) = \{\vec{r} \in \mathbb{N}^p : |\vec{r}| = r\}$ *and the unit eigenfunctions are*

$$\begin{cases} \mathsf{H}^{(\vec{r})} = \mathrm{span}\left\{Y_{\vec{r},\vec{l}}(\frac{\cdot}{\sqrt{d_0}})\right\}_{\vec{l}\in[B(d_0,\vec{r})]} \\ \dim(\mathsf{H}^{(r)}) \sim d^r \quad \textit{and} \quad \lambda(\mathsf{H}^{(\vec{r})}) \sim d^{-r} k^{(r)}(0) \quad \textit{if} \quad k^{(r)}(0) \neq 0 \end{cases} \tag{17}$$

(2)+Locality: $\mathcal{K} = \mathcal{K}_{\mathsf{VEC}}$. $Q(\mathcal{K}, r) = \{\vec{r} \in B(\mathbb{N}^p, s) : |\vec{r}| = r\}$ *the unit eigenfunctions are*

$$\begin{cases} \mathsf{H}_{\mathsf{VEC}}^{(\vec{r})} = \mathrm{span}\left\{Y_{\vec{r},\vec{l}}(\frac{\cdot}{\sqrt{d_0}})\right\}_{\vec{l}\in[B(d_0,\vec{r})]} \\ \dim(\mathsf{H}_{\mathsf{VEC}}^{(r)}) \sim ps^{r-1}d_0^r = s^{r-1}d^{1-\alpha+r\alpha} \quad \textit{and} \quad \lambda(\mathsf{H}_{\mathsf{VEC}}^{(\vec{r})}) \sim p^{-1}(sd_0)^{-r}k^{(r)}(0) \quad \textit{if} \quad k^{(r)}(0) \neq 0 \end{cases} \tag{18}$$

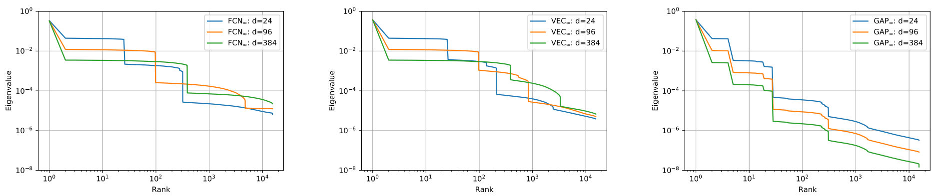

Figure 5: **Eigenvalue Decay of Relu NTK of** FCN_∞**,** VEC_∞ **and** GAP_∞**.** $d_0 = s = 3$. The eigenvalues of GAP_∞ decays *faster* because with $m = 15k$ many samples, higher order eigenspace can be covered by GAP_∞ but not $\mathsf{FCN}_\infty/\mathsf{VEC}_\infty$ due to Theorem 4.1.

(3)+Locality + Shifting: $\mathcal{K} = \mathcal{K}_{\mathsf{GAP}}$**.** $Q(\mathcal{K}, r) = \{\vec{r} \in B(\mathbb{N}^p, s)/\tau : |\vec{r}| = r\}$*, the unit eigenfunctions are*

$$\begin{cases} \mathsf{H}_{\mathsf{GAP}}^{(\vec{r})} = \mathrm{span}\left\{\frac{1}{\sqrt{p}} \sum_{\tau \in [p]} Y_{\vec{r},\vec{l}}(\frac{\tau}{\cdot}\sqrt{d_0})\right\}_{\vec{l} \in [B(d_0, \vec{r})]} \\ \dim(\mathsf{H}_{\mathsf{GAP}}^{(r)}) \sim s^{r-1} d_0^r \quad \textit{and} \quad \lambda(\mathsf{H}_{\mathsf{GAP}}^{(\vec{r})}) \sim p^{-1}(sd_0)^{-r} k^{(r)}(0) \quad \textit{if} \quad k^{(r)}(0) \neq 0 \end{cases} \tag{19}$$

5 Related Work

The study of infinite networks dates back to the seminal work by Neal [8] who showed the convergence of single hidden-layer networks to Gaussian Processes (GPs). Recently, there has been renewed interest in studying random, infinite, networks starting with concurrent work on "conjugate kernels" [10, 35] and "mean-field theory" [9, 36], taking a statistical learning and statistical physics view of points, resp. Since then this analysis has been extended to include a wide range for architectures [20, 21, 37, 29, 26, 38]. The inducing kernel is often referred to as the Neural Network Gaussian Process (NNGP) kernel. The neural tangent kernel (NTK), first introduced in Jacot et al. [22], along with followup work [12, 39] showed that the distribution of functions induced by gradient descent for infinite-width networks is a Gaussian Process with NTK as the kernel.

The study of implicit bias (regularization) of gradient descent has received considerable interests. The work [15, 40–43] demonstrate the convergence of SGD to the maximal margin solution for logistic-type losses during late time training. [44–50] study the early-time SGD dynamics, spectral biases of neural networks. These results aim to explain the order of learning of neural networks: functions of less complexity are usually learned before more complex functions.

[27] is the first to show that the prediction functions obtained from training FCN depend, in addition on the labels, only on the covariance of the input data. This implies our result regarding the $\mathrm{O}(3d)$ invariance of FCN. By utilizing this symmetry, recent work [51] constructs a particular task where the label function is a second order polynomial of the inputs and show that orthogonal invariance algorithm requires sample size of order d^2 while there is a convnet requires only $O(1)$ samples. Their convnet essentially corresponds to the $d_0 = s = 1$ and $r = 2$ case of Theorem 4.1, in which the dimension of this eigenspace (and indeed of all r-eigenspace by treating r as a finite constant as $d \to \infty$) of GAP_∞ is $O(1)$ while the dimension of the 2-eigenspace of FCN_∞ is of order d^2. See Subsection. E.4.

6 Conclusion

In this paper, we consider machine learning methods as an integrated system of data, models and inference algorithms and study the basic symmetries of various machine learning systems. We surface the importance of locality in modern machine learning systems through large scale empirical study and through an eigendecomposition of one-layer infinite networks. However, we haven't addressed two import questions (1) theoretical characterization of the effect of composing locality and (2) the mathematical understanding of DIDE and how the prior is corrected by the coupled effect of data and gradient descent. We leave them to future work.

Checklist

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

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

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

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

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[5] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[6] 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 arXiv:2001.08361*, 2020.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Radford M. Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

[9] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, 2016.

[10] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, 2016.

[11] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[12] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

[13] Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[14] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks. *arXiv preprint arXiv:1806.00468*, 2018.

[15] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data, 2018.

[16] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[17] Alex J Smola, Zoltan L Ovari, Robert C Williamson, et al. Regularization with dot-product kernels. *Advances in neural information processing systems*, pages 308–314, 2001.

[18] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[19] Yann Lecun. Generalization and network design strategies. In *Connectionism in perspective*. Elsevier, 1989.

[20] Jaehoon Lee, Yasaman Bahri, Roman Novak, Sam Schoenholz, Jeffrey Pennington, and Jascha Sohl-dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[21] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

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

[23] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

[24] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, 2018.

[25] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

[26] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[27] Neha S. Wadia, Daniel Duckworth, Samuel S. Schoenholz, Ethan Dyer, and Jascha Sohl-Dickstein. Whitening and second order optimization both destroy information about the dataset, and can make generalization impossible. *arxiv preprint arXiv:2008.07545*, 2020.

[28] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402, 2018.

[29] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[30] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

[31] 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. *arXiv preprint arXiv:1912.02803*, 2019.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[33] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[34] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1Ddp1-Rb`.

[35] Amit Daniely. SGD learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems 30*. 2017.

[36] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *International Conference on Learning Representations*, 2017.

[37] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, 2018.

[38] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks, 2020.

[39] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[40] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJeLIgBKPS`.

[41] Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798, 2019.

[42] Ziwei Ji and Matus Jan Telgarsky. Gradient descent aligns the layers of deep linear networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[43] Lénaïc Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 1305–1338. PMLR, 09–12 Jul 2020. URL http://proceedings.mlr.press/v125/chizat20a.html.

[44] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *arXiv preprint arXiv:1905.11604*, 2019.

[45] Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *arXiv preprint arXiv:2006.14599*, 2020.

[46] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[47] Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

[48] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

[49] Lili Su and Pengkun Yang. On learning over-parameterized neural networks: A functional approximation perspective. *arXiv preprint arXiv:1905.10826*, 2019.

[50] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*, 2019.

[51] Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets? *arXiv preprint arXiv:2010.08515*, 2020.

[52] Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

[53] Mei Song, Andrea Montanari, and P Nguyen. A mean field view of the landscape of two-layers neural networks. *Proceedings of the National Academy of Sciences*, 115:E7665–E7671, 2018.

[54] Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

[55] Christopher Frye and Costas J. Efthimiou. Spherical harmonics in p dimensions. 2012.

[56] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *arXiv preprint arXiv:2006.13409*, 2020.

Supplementary Material

A Glossary

We use the following abbreviations in this work:

- $+\mathsf{L2}$: Adding L2 regularization.
- $+\mathsf{LR}$: Using a large learning rate.
- $+\mathsf{DA}$: Applying MixUp data augmentation.
- $+\mathsf{C}$: Centering the outputs of the network.
- $-\mathsf{C}$: Remove centering.
- FCN_n:Fully-connected networks with width n.
- FCN_∞: Infinite width FCN_n.
- VEC_n: Convnet with width n and a flattening readout layer.
- VEC_∞:Infinite width VEC_n.
- LCN_n: Locally-connected network with width n.
- LCN_∞: Infinite width LCN_n, which is the samme as VEC_∞.
- GAP_n: Convnet with width n and a global average readout layer.
- GAP_∞:Infinite width GAP_n.
- LAP_n^k: Similar to GAP_n, except the readout layer is a (k,k) average pooling.
- LAP_∞^k: Infinite width LAP_n^k.

B Proof of Theorem 2.1

We use FCN_n to denote the class of functions that can be expressed by L-hidden layer fully-connected networks whose widths are equal to n. Similar notation applies to other architectures.

Corollary 1. *We have the following*

$$\mathsf{GAP}_n \subseteq \mathsf{VEC}_n \subseteq \mathsf{LCN}_n \subseteq \mathsf{VEC}_{dn}, \quad \mathsf{LCN}_n \subseteq \mathsf{FCN}_{dn} \tag{S1}$$

Proof. We only need to prove $\mathsf{LCN}_n \subseteq \mathsf{VEC}_{dn}$ because the others are obvious. Let $\mathsf{LCN}_n(x)^l_{\alpha,i}$ denote the post-activation at layer l, spatial location α and channel index i of a LCN_n with input x and $\mathsf{VEC}_n(x)^l_{\alpha,i}$ is defined similarly. It suffices to prove that for any LCN with width n there is a VEC with width dn such that for any $l \geq 1$ (i.e. not the input layer)

$$\mathsf{VEC}_{dn}(x)^l_{\alpha,\alpha n+i} = \mathsf{LCN}_n(x)^l_{\alpha,i} \tag{S2}$$

since we could choose the readout weights of VEC_{dn} at locations $(\alpha, \alpha n + i)$ to match the one of LCN_n at locaton (α, i) and zero out the remaining entries. We prove this by induction and assume it holds for l (the base case $l = 1$ is obvious). Then the LCN_n and VEC_n at layer $l+1$ can be written as

$$\mathsf{LCN}_n(x)^{l+1}_{\alpha,j} = \phi\left(\frac{1}{\sqrt{n(2k+1)}} \sum_{i\in[n],\beta\in[-k,k]} \mathsf{LCN}_n(x)^l_{\alpha+\beta,i}\,\omega^{l+1}_{\beta,ij}(\alpha)\right)$$

and

$$\mathsf{VEC}_{dn}(x)^{l+1}_{\alpha,j} = \phi\left(\frac{1}{\sqrt{dn(2k+1)}} \sum_{i\in[dn],\beta\in[-k,k]} \mathsf{VEC}_{dn}(x)^l_{\alpha+\beta,i}\,\tilde{\omega}^{l+1}_{\beta,ij}\right)$$

One can show that Equation S2 holds for $(l+1)$ by choosing the parameters of VEC_{dn} as follows

$$\tilde{\omega}^{l+1}_{\beta,ij} = \sqrt{d}\,\omega^{l+1}_{\beta,i-(\alpha+\beta)n,j-\alpha n} \quad \text{if} \quad \alpha n \leq j < \alpha(n+1) \quad \text{and} \quad (\alpha+\beta)n \leq i < (\alpha+\beta)(n+1)$$

and 0 otherwise.

$\square$

C Proof of Symmetries

Proof. For simplicity, we present the proof for full-batch training. The proof applies to mini-batch training as long as order of the mini-batch is fixed. Let τ be a rotation in $\mathrm{O}(3d)$ or $\mathrm{O}(3)^d$ or $\mathrm{O}(3) \otimes \mathbf{I}_d$, depending on the architectures ($\mathsf{FCN}_n, \mathsf{LCN}_n, \mathsf{VEC}_n, \mathsf{GAP}_n$) and the tuple θ and γ denote the parameters of the first and remaining layers of the network, resp. Let $h(\tau x, \theta) = \langle \tau x, \theta \rangle$ denote the pre-activations of the first-hidden layer in the rotated coordinate. Here $\langle \cdot, \cdot \rangle$ is the bilinear map (a dense layer or a convolutional layer with or without weight-sharing, etc.), not the inner product. The loss with L^2-regularization is

$$R_\lambda(\theta, \gamma) = L(h(\tau \mathcal{X}, \theta), \gamma) + \frac{1}{2}\lambda(\|\theta\|_2^2 + \|\gamma\|_2^2) \tag{S3}$$

where $L(h(\tau \mathcal{X}, \theta), \gamma)$ is the raw loss of the network. For each random instantiation $\theta = \theta_0$ with θ_0 drawn from standard Gaussian iid, we instantiate a coupled network from the un-rotated coordinates but with a different instantiation in the first layer $\theta^\tau = \tau^* \theta_0$ and keep the remaining layers unchanged, i.e. $\gamma^\tau = \gamma_0$. Here τ^* is the adjoint of τ and note that $\tau^*\theta_0$ and θ_0 have the same distribution by the Gaussian initialization of θ_0 and the definition of τ. The regularized loss associated to this instantiation is

$$R_\lambda(\theta^\tau, \gamma^\tau) = L(h(\mathcal{X}, \theta^\tau), \gamma^\tau) + \frac{1}{2}\lambda(\|\theta^\tau\|_2^2 + \|\gamma^\tau\|_2^2) \tag{S4}$$

It suffices to prove that for each instantiation $\theta = \theta_0$ drawn from Gaussian, the following holds for all gradient steps t

$$(\theta_t^\tau, \gamma_t^\tau) = (\tau^*\theta_t, \gamma_t). \tag{S5}$$

We prove this by induction on t and $t = 0$ is true by definition. Assume it holds when $t = t$. Now the update in γ and γ^τ with learning rate η are

$$\gamma_{t+1} = \gamma_t - \eta \left(\left.\frac{\partial L}{\partial \gamma}\right|_{(h(\tau\mathcal{X}, \theta_t), \gamma_t)} \right)^T - \eta\lambda\gamma_t \tag{S6}$$

$$\gamma_{t+1}^\tau = \gamma_t^\tau - \eta \left(\left.\frac{\partial L}{\partial \gamma}\right|_{(h(\mathcal{X}, \theta_t^\tau), \gamma_t^\tau)} \right)^T - \eta\lambda\gamma_t^\tau \tag{S7}$$

It is clear $\gamma_{t+1} = \gamma_{t+1}^\tau$ by induction since $h(\tau\mathcal{X}, \theta_t) = h(\mathcal{X}, \theta_t^\tau)$. Similarly,

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\partial L}{\partial h} \left.\frac{\partial h}{\partial \theta}\right|_{(\tau\mathcal{X}, \theta_t))} \right)^T - \lambda\theta_t \tag{S8}$$

$$\theta_{t+1}^\tau = \theta_t^\tau - \eta \left(\frac{\partial L}{\partial h} \left.\frac{\partial h}{\partial \theta^\tau}\right|_{(\mathcal{X}, \theta_t^\tau)} \right)^T - \lambda\theta^\tau \tag{S9}$$

Note that by the chain rule and induction assumption

$$\left.\frac{\partial h}{\partial \theta^\tau}\right|_{(\mathcal{X}, \theta_t^\tau)} = \left.\frac{\partial h}{\partial \theta}\right|_{(\mathcal{X}, \theta_t^\tau)} \frac{\partial \theta^\tau}{\partial \theta} = \left.\frac{\partial h}{\partial \theta}\right|_{(\mathcal{X}, \theta_t^\tau)} \tau \tag{S10}$$

This implies $\theta_{t+1}^\tau = \tau^*\theta_{t+1}$.

□

Remark S1. It is not difficult to see the apply proof apply to Non-Gaussian i.i.d. initialization (e.g. uniform distribution) and/or adding L^p-regularization when the rotation groups are replaced by the corresponding permutation groups. Empirically, we observe that replacing the first layer Gaussian initialization by uniform distribution does not change the performance of the network much. See Fig.S2.

Remark S2. The proof works for all parameterization methods, including NTK-parameterization[11], standard parameterization [52], mean-field parameterization[53] and ABC-parameterization [54]

D Eigendecomposition of Infinite Networks

To gain insight into the inductive biases of various architectures, we eigendecompose the kernels using spherical harmonics. We assume the input space $\mathcal{X} = \{\xi = (\xi_0, \dots, \xi_{p-1}) \in (\sqrt{d_0}\mathbb{S}^{(d_0-1)})^p\} \subseteq \mathbb{R}^{d_0 p}$, i.e. the p-product of (d_0-1)-sphere with radius $\sqrt{d_0}$. We call $\xi_i \in \sqrt{d_0}\mathbb{S}^{(d_0-1)}$ a mini-patch and $(\xi_i, \xi_{i+1}, \dots, \xi_{i+s-1}) \in (\sqrt{d_0}\mathbb{S}^{(d_0-1)})^s\}$ a patch for $i \in [p]$, where circular boundary condition is assumed. We consider the asymptotic limit when $d_0 = d^\alpha, p = d^{1-\alpha}$ and $d = pd_0 \to \infty$ and treat $0 < \alpha < 1$ and s as fixed constant.

The input space $\mathcal{X}$ is associated with the product measure $\mu \equiv \sigma_{d_0}^p$, where σ_{d_0} is the normalized uniform measure on $\sqrt{d_0}\mathbb{S}^{(d_0-1)}$. The kernels associated to the one-hidden layer infinite networks (NNGP and NTK) have the following general forms

$$\mathcal{K}_{\mathsf{FCN}}(\xi, \eta) = k\left(\frac{1}{p}\sum_{i\in[p]} \xi_i^T \eta_i / d_0\right), \tag{S11}$$

$$\mathcal{K}_{\mathsf{VEC}}(\xi, \eta) = \frac{1}{p}\sum_{i\in[p]} k\left(\frac{1}{s}\sum_{b\in[s]} \xi_{i+b}^T \eta_{i+b} / d_0\right), \tag{S12}$$

$$\mathcal{K}_{\mathsf{GAP}}(\xi, \eta) = \frac{1}{p^2}\sum_{i,j\in[p]} k\left(\frac{1}{s}\sum_{b\in[s]} \xi_{i+b}^T \eta_{j+b} / d_0\right), \tag{S13}$$

for FCN_∞, VEC_∞ and GAP_∞ resp. Note that the exact form of the (positive definite) kernel function $k : \mathbb{R} \to \mathbb{R}$ depends on the type of the kernels (NNGP vs NTK), activations, hyperparameters and etc. We assume the kernel is sufficiently smooth in $(-1, 1)$ and the Tayor expansion of $k^{(r)}$ converges uniformly in $[-1, 1]$ for sufficiently many $r \in \mathbb{N}$. We use the notation that $A \sim B$ if there are positive constants c and C such that $cA \leq B \leq CA$ for d sufficiently large. We use $\mathcal{K}$ to represent any kernels above and consider it as a Hilbert–Schmidt operator on $L^2(\mathcal{X}, \mu)$

$$\mathcal{K}f(\xi) = \int_{\mathcal{X}} \mathcal{K}(\xi, \eta) f(\eta) d\mu, \quad f \in L^2(\mathcal{X}, \mu), \tag{S14}$$

which is well-defined since μ is a probability measure and k is bounded.

D.1 Legendre Polynomials, Spherical Harmonics and their Tensor Products.

Our notation follows closely from [55], an excellent introduction to spherical harmonics.

Legendre Polynomials. Let ω_{d_0} be the measure defined on the interval $I = [-1, 1]$

$$\omega_{d_0}(t) = (1 - t^2)^{(d_0-3)/2} \tag{S15}$$

The Legendre polynomials[2] $\{P_r(t) : r \in \mathbb{N}\}$ is an orthogonal basis for the Hilbert space $L^2(I, \omega_{d_0})$, i.e.

$$\int_I P_r(t) P_{r'}(t) \omega_{d_0}(t) dt = 0 \quad \text{if} \quad r \neq r' \tag{S16}$$

Here $P_r(t)$ is a degree r polynomials with $P_r(1) = 1$ and satisfies the Rodrigues formula

Lemma 1 (Rodrigues Formula. Proposition 4.19 [55])**.**

$$P_r(t) = c_r \omega_{d_0}^{-1} \left(\frac{d}{dt}\right)^r (1 - t^2)^{r + (d_0-3)/2}, \tag{S17}$$

where

$$c_r = \frac{(-1)^r}{2^r (r + (d_0-3)/2)_r} \tag{S18}$$

[2] More accurate, this should be called Gegenbauer Polynomials. However, we decide to stick to the terminology in [55]

In the above lemma, $(x)_l$ denotes the falling factorial

$$(x)_l \equiv x(x-1)\cdots(x-l+1) \tag{S19}$$

$$(x)_0 \equiv 1 \tag{S20}$$

Spherical Harmonics. Let $d\mathbb{S}_{d_0-1}$ define the (un-normalized) uniform measure on the unit sphere $\mathbb{S}_{d_0-1}$. Then

$$|\mathbb{S}_{d_0-1}| \equiv \int_{\mathbb{S}_{d_0-1}} d\mathbb{S}_{d_0-1} = \frac{2\pi^{d_0/2}}{\Gamma(\frac{d_0}{2})} \tag{S21}$$

The normalized measure on this sphere is defined to be

$$d\sigma_{d_0} = \frac{1}{|\mathbb{S}_{d_0-1}|} d\mathbb{S}_{d_0-1} \quad \text{and} \quad \int_{\mathbb{S}_{d_0-1}} d\sigma_{d_0} = 1 \tag{S22}$$

The spherical harmonics $\{Y_{r,l}\}_{r,l}$ in $\mathbb{R}^{d_0}$ are homogeneous harmonic polynomials that form an orthonormal basis in $L^2(\mathbb{S}_{d_0-1}, \sigma_{d_0})$

$$\int_{\xi\in\mathbb{S}_{d_0-1}} Y_{r,l}(\xi) Y_{r',l'}(\xi) d\sigma_{d_0} = \delta_{(r,l)=(r',l')}. \tag{S23}$$

Here $Y_{r,l}$ denotes the l-th spherical harmonic whose degree is r, where $r \in \mathbb{N}$, $l \in [N(d_0, r)]$ and

$$N(d_0, r) = \frac{2r + d_0 - 2}{r} \binom{d_0 + r - 3}{r - 1} \sim d_0^r/r! \quad \text{as} \quad d_0 \to \infty\,. \tag{S24}$$

The Legendre polynomials and spherical harmonics are related through the addition theorem.

Lemma 2 (Addition Theorem. Theorem 4.11 [55])**.**

$$P_r(\xi^T\eta) = \frac{1}{N(d_0, r)} \sum_{l\in[N(d_0,r)]} Y_{r,l}(\xi) Y_{r,l}(\eta), \qquad \xi, \eta \in \mathbb{S}_{d_0-1}\,. \tag{S25}$$

Tensor Products. Let $p \in \mathbb{N}$, $\vec{r} \in \mathbb{N}^p$, $I^p = [-1, 1]^p$ and $\omega_{d_0}^p$ be the product measure on I^p. Then the (product of) Legendre polynomials

$$P_{\vec{r}}(\vec{t}) = \prod_{j\in[p]} P_{r_j}(t_j)\,, \quad \vec{t} = (t_1, \ldots, t_p) \in I^p \tag{S26}$$

form an orthogonal basis for the Hilbert space $L^2(I^p, \omega_{d_0}^p) = (L^2(I, \omega_{d_0})^) \otimes p$. Similarly, the product of spherical harmonics

$$Y_{\vec{r},\vec{l}} = \prod_{j\in[p]} Y_{r_j,l_j}, \quad \vec{l} = (l_1, \ldots, l_p) \in [N(d_0, \vec{r})] \equiv \prod_{j\in p}[N(d_0, r_j)] \tag{S27}$$

form an orthonormal basis for the product space

$$L^2(\mathbb{S}_{d_0-1}^p, \sigma_{d_0}^p) = (L^2(\mathbb{S}_{d_0-1}, \sigma_{d_0}))^{\otimes p}. \tag{S28}$$

Elements in the set $\{Y_{\vec{r},\vec{l}}\}_{\vec{l}\in[B(d_0,\vec{r})]}$ are called degree (order) $\vec{r}$ spherical harmonics in $L^2(\mathbb{S}_{d_0-1}^p, \sigma_{d_0}^p)$ and also degree r spherical harmonics if $|\vec{r}| = r \in \mathbb{N}$.

D.2 "Fourier" Decomposition.

Let $K(\vec{t}) \in L^2(\mathbb{S}_{d_0-1}^p, \sigma_{d_0}^p)$. Then we have the following "Fourier decomposition" (the convergence is in L^2),

$$K(\vec{t}) = \sum_{\vec{r}\in\mathbb{N}^p} \hat{K}(\vec{r}) P_{\vec{r}}(\vec{t}) \tag{S29}$$

where the "Fourier coefficients" are

$$\hat{K}(\vec{r}) = \langle K, P_{\vec{r}} \rangle_{L^2(I^p,\omega_{d_0}^p)} / \langle P_{\vec{r}}, P_{\vec{r}} \rangle_{L^2(I^p,\omega_{d_0}^p)}\,. \tag{S30}$$

Applying Lemma 2, we have the harmonic decomposition (the convergence is in L^2)

$$K(\xi^T\eta) = \sum_{\vec{r}\in\mathbb{N}^p} \hat{K}(\vec{r})N(d_0,\vec{r})^{-1} \sum_{\vec{l}\in[B(d_0,\vec{r})]} Y_{\vec{r},\vec{l}}(\xi)Y_{\vec{r},\vec{l}}(\eta), \quad \xi,\eta\in\mathbb{S}^p_{d_0-1} \tag{S31}$$

Clearly, as an integral operator

$$\int_{\mathbb{S}^p_{d_0-1}} K(\xi^T\eta)Y_{\vec{r},\vec{l}}(\eta)d\sigma^p_{d_0} = \hat{K}(\vec{r})N(d_0,\vec{r})^{-1}Y_{\vec{r},\vec{l}}(\xi)\,. \tag{S32}$$

Theorem D.1. *Let $K(\vec{t})\in L^2(\mathbb{S}^p_{d_0-1},\sigma^p_{d_0})$. Then.*

$$K(\xi^T\eta) = \sum_{\vec{r}\in\mathbb{N}^p} \hat{K}(\vec{r})N(d_0,\vec{r})^{-1} \sum_{\vec{l}\in[B(d_0,\vec{r})]} Y_{\vec{r},\vec{l}}(\xi)Y_{\vec{r},\vec{l}}(\eta), \quad \xi,\eta\in\mathbb{S}^p_{d_0-1} \tag{S33}$$

If in addition $\|K\|_{C^{|\vec{r}|+1}(I^p)}<\infty$, then

$$\hat{K}(\vec{r}) = \vec{r}!^{-1}\left(K^{(\vec{r})}(0) + \mathcal{O}(\|K\|_{C^{|\vec{r}|+1}(I^p)}pd_0^{-\frac{1}{2}})\right)\,. \tag{S34}$$

Therefore, the eigenvalues of $K(\xi^T\eta)$ are $\hat{K}(\vec{r})N(d_0,\vec{r})^{-1}$, with eigenspace spanned by the (unit) eigenvectors $\{Y_{\vec{r},\vec{l}}\}_{\vec{l}\in[N(d_0,\vec{r})]}$ whose dimension is $N(d_0,\vec{r})$, resp.

D.3 Eigendecomposing the Infinite Networks

To handle the patch, we introduce the s-banded subset of $\mathbb{N}^p$. For $i,j\in[p]$, define the a distance in the cyclic group $[p]=\mathbb{Z}/p\mathbb{Z}$ to be

$$\mathrm{dist}(i,j) = \min\{|i-j|, p-|i-j|\},$$

and the diameter of $\vec{r}\in\mathbb{N}^p$ to be

$$\mathrm{diam}(\vec{r}) = \mathrm{dist}(\mathrm{argmax}_j r_j\neq 0, \mathrm{argmin}_j r_j \neq 0) \tag{S35}$$

The s-banded subset of $\mathbb{N}^p$ is the collection of points whose diameter is less than s, i.e.,

$$B(\mathbb{N}^p,s) = \{\vec{r}\in\mathbb{N}^p : \text{diam}(r)\le s-1\} \tag{S36}$$

This implies $Y_{\vec{r},\vec{l}}$ is a function defined on a patch if and only if $\vec{r}\in B(\mathbb{N}^p,s)$.

Let τ be shifting operator $\tau\vec{r}=(r_{p-1},r_0,\dots,r_{p-2})$, where $\vec{r}=(r_0,\dots,r_{p-1})\in\mathbb{N}^p$. The quotient space $B(\mathbb{N}^p,s)/\tau$ denotes a subset of $B(\mathbb{N}^p,s)$ by identifying $\vec{v}=\vec{v}'$ as the same element if $\vec{v}=\tau^a\vec{r}'$ for some $a\in[p]$.

In deep learning, it is more convenient to work on the non-unit sphere $\sqrt{d_0}\mathbb{S}_{d_0-1}$. We still use σ_{d_0} to denote the normalized (probability) measure on $\sqrt{d_0}\mathbb{S}_{d_0-1}$. The spherical harmonics with unit norms are

$$Y_{r_j,l_j}\left(\frac{\xi_j}{\sqrt{d_0}}\right)\in L^2\left(\sqrt{d_0}\mathbb{S}^{(d_0-1)},\sigma_{d_0}\right) \tag{S37}$$

$$Y_{\vec{r},\vec{l}}\left(\frac{\xi}{\sqrt{d_0}}\right)\in L^2\left(\left(\sqrt{d_0}\mathbb{S}^{(d_0-1)}\right)^p,\sigma^p_{d_0}\right) \tag{S38}$$

The following theorem characterize the inductive biases induced by locality and symmetry (i.e. shifting invariant) for infinite networks. It shows that locality (VEC_∞) dramatically reduces both the dimensions of $r\ge 1$ eigenspaces and the spectral gap among them. In addition, pooling (i.e. resulting shifting invariant for GAP_n) reduces their dimensions by an additional factor of p. See Sec.E for the implication of this theorem to learning.

Theorem D.2. *We have the following eigendecomposition for the integral operator $\mathcal{K}$*

$$\mathsf{H} = \bigcup_{r\in\mathbb{N}} \mathsf{H}^{(r)} = \bigcup_{r\in\mathbb{N}}\bigcup_{\vec{r}\in Q(\mathcal{K},r)} \mathsf{H}^{(\vec{r})}, \tag{S39}$$

where $Q(\mathcal{K},r)$ is a quantifier defined below. If $r=0$, $\mathsf{H}^{(0)}$ is the space of constant functions and the eigenvalue is $\sim k(0)$. For $r\ge 1$, we have the following.

(1)Base Case: $\mathcal{K} = \mathcal{K}_{\mathsf{FCN}}$**.** $Q(\mathcal{K}, r) = \{\vec{r} \in \mathbb{N}^p : |\vec{r}| = r\}$ *and the unit eigenfunctions are*

$$\begin{cases} \mathsf{H}^{(\vec{r})} = \operatorname{span}\left\{ Y_{\vec{r},\vec{l}}(\frac{\cdot}{\sqrt{d_0}}) \right\}_{\vec{l} \in [B(d_0,\vec{r})]} \\ \dim(\mathsf{H}^{(r)}) \sim d^r \quad \textit{and} \quad \lambda(\mathsf{H}^{(\vec{r})}) \sim d^{-r}\delta(k^{(r)}(0)) \end{cases} \tag{S40}$$

(2)+Locality: $\mathcal{K} = \mathcal{K}_{\mathsf{VEC}}$**.** , $Q(\mathcal{K}, r) = \{\vec{r} \in B(\mathbb{N}^p, s) : |\vec{r}| = r\}$ *the unit eigenfunctions are*

$$\begin{cases} \mathsf{H}^{(\vec{r})}_{\mathsf{VEC}} = \operatorname{span}\left\{ Y_{\vec{r},\vec{l}}(\frac{\cdot}{\sqrt{d_0}}) \right\}_{\vec{l} \in [B(d_0,\vec{r})]} \\ \dim(\mathsf{H}^{(r)}_{\mathsf{VEC}}) \sim ps^{r-1}d_0^r = s^{r-1}d^{1-\alpha+r\alpha} \quad \textit{and} \quad \lambda(\mathsf{H}^{(\vec{r})}_{\mathsf{VEC}}) \sim p^{-1}(sd_0)^{-r}\delta(k^{(r)}(0)) \end{cases} \tag{S41}$$

(3)+Locality + Shifting: $\mathcal{K} = \mathcal{K}_{\mathsf{GAP}}$**.** $Q(\mathcal{K}, r) = \{\vec{r} \in B(\mathbb{N}^p, s)/\tau : |\vec{r}| = r\}$*, the unit eigenfunctions are*

$$\begin{cases} \mathsf{H}^{(\vec{r})}_{\mathsf{GAP}} = \operatorname{span}\left\{ \frac{1}{\sqrt{p}} \sum_{\tau \in [p]} Y_{\vec{r},\vec{l}}(\frac{\tau}{\cdot}\sqrt{d_0}) \right\}_{\vec{l} \in [B(d_0,\vec{r})]} \\ \dim(\mathsf{H}^{(r)}_{\mathsf{GAP}}) \sim (sd_0)^r = s^r d^{r\alpha} \quad \textit{and} \quad \lambda(\mathsf{H}^{(\vec{r})}_{\mathsf{GAP}}) \sim p^{-1}(sd_0)^{-r}\delta(k^{(r)}(0)) \end{cases} \tag{S42}$$

Proof. Our main tool is Theorem D.1.

Base Case $\mathcal{K}_{\mathsf{FCN}}$**.** Setting

$$K(\vec{t}) = k\left(\frac{1}{p}\sum_{j \in [p]} t_j\right) \tag{S43}$$

and applying Theorem D.1 give

$$K(\xi^T\eta/d_0) = \sum_{\vec{r} \in \mathbb{N}^p} \hat{K}(\vec{r}) N(d_0, \vec{r})^{-1} \sum_{\vec{l} \in [B(d_0,\vec{r})]} Y_{\vec{r},\vec{l}}(\xi/\sqrt{d_0}) Y_{\vec{r},\vec{l}}(\eta/\sqrt{d_0}) \tag{S44}$$

for $\xi/\sqrt{d_0}$ and $\eta/\sqrt{d_0} \in \mathbb{S}^p_{d_0-1}$. By the chain rule

$$\begin{aligned} \hat{K}(\vec{r}) &= \vec{r}!^{-1}\left(K^{(\vec{r})}(0) + \mathcal{O}(\|K\|_{C^{|\vec{r}|+1}(I^p)} p d_0^{-\frac{1}{2}})\right) \\ &= \vec{r}!^{-1}\left(p^{-\vec{r}} k^{(\vec{r})}(0) + \mathcal{O}(p^{-|\vec{r}|-1}\|k\|_{C^{|\vec{r}|+1}(I)} p d_0^{-\frac{1}{2}})\right) \\ &= \vec{r}!^{-1} p^{-|\vec{r}|}\left(k^{(\vec{r})}(0) + \mathcal{O}(d_0^{-\frac{1}{2}})\right) \end{aligned}$$

As $d_0 \to \infty$, if $k^{(|\vec{r}|)}(0) \neq 0$ then the eigenvalue of the $\vec{r}$-eigenspace is

$$\lambda(\mathsf{H}^{(\vec{r})}) \sim k^{(|\vec{r}|)}(0)\vec{r}!^{-1} p^{-|\vec{r}|} N(d_0, \vec{r})^{-1} \sim (pd_0)^{-|\vec{r}|} = d^{-|\vec{r}|} \tag{S45}$$

The dimension is $N(d_0, \vec{r}) \sim d_0^{|\vec{r}|}/\vec{r}!$. This completes the proof of the base case.

+Locality $\mathcal{K}_{\mathsf{VEC}}$. Recall that

$$\mathcal{K}_{\mathsf{VEC}}(\xi, \eta) = \frac{1}{p}\sum_{i \in [p]} k\left(\frac{1}{s}\sum_{b \in [s]} \xi_{i+b}^T \eta_{i+b}/d_0\right), \tag{S46}$$

which is a sum of kernels supported on patches. Setting

$$K(t_1, \ldots, k_s) = k\left(\frac{1}{s}\sum_{j \in [s]} t_j\right), \tag{S47}$$

applying Theorem D.1 with $p = s$ to each summand implies

$$\mathcal{K}_{\mathsf{VEC}}(\xi, \eta) = \frac{1}{p}\sum_{i \in [p]}\sum_{\vec{r} \in \mathbb{N}^s} \hat{K}(\vec{r}) N(d_0, \vec{r})^{-1} \sum_{\vec{l}} Y_{\vec{r},\vec{l}}(\xi_{i:i+s}/\sqrt{d_0}) Y_{\vec{r},\vec{l}}(\eta_{i:i+s}/\sqrt{d_0}) \tag{S48}$$

$$= \sum_{\vec{r} \in \mathbb{N}^s} \frac{1}{p} \hat{K}(\vec{r}) N(d_0, \vec{r})^{-1} \sum_{\vec{l}} \sum_{i \in [p]} Y_{\vec{r},\vec{l}}(\xi_{i:i+s}/\sqrt{d_0}) Y_{\vec{r},\vec{l}}(\eta_{i:i+s}/\sqrt{d_0}) \tag{S49}$$

in which we have applied the Fubini Theorem. Similarly, if $k^{(\vec{r})}(0) \neq 0$, the term

$$\hat{K}(\vec{r})N(d_0,\vec{r})^{-1} \sim k^{(\vec{r})}(0)(sd_0)^{-|\vec{r}|}, \tag{S50}$$

where the $s^{-|\vec{r}|}$ is coming from applying the chain rule to Equation S47. Next, we treat the functions $Y_{\vec{r},\vec{l}}(\xi_{i:i+s}/\sqrt{d_0})$ defined on a patch as functions $Y_{\vec{r},\vec{l}}(\xi/\sqrt{d_0})$ defined on the whole space $(\sqrt{d_0}\mathbb{S}_{d_0-1})^p$ by restricting $\vec{r} \in B(\mathbb{N}^p, s)$. As such we need to count, for a given $\vec{r}$, the number of patches the function $Y_{\vec{r},\vec{l}}(\xi_{i:i+s}/\sqrt{d_0})$ belong to, which turns out to be $(s - \mathrm{diam}(\vec{r}))$. We could reorder the terms in $\mathcal{K}_{\mathsf{VEC}}$ as follows

$$\mathcal{K}_{\mathsf{VEC}}(\xi,\eta) = \sum_{\vec{r}\in B(\mathbb{N}^p,s)} \frac{1}{p}\hat{K}(\vec{r})N(d_0,\vec{r})^{-1}(s-\mathrm{diam}(\vec{r}))\sum_{\vec{l}} Y_{\vec{r},\vec{l}}(\xi/\sqrt{d_0})Y_{\vec{r},\vec{l}}(\eta/\sqrt{d_0}) \tag{S51}$$

Clearly, $Y_{\vec{r},\vec{l}}(\xi/\sqrt{d_0})$ are the eigenfunctions of unit norm with eigenvalues

$$p^{-1}\hat{K}(\vec{r})N(d_0,\vec{r})^{-1}(s-\mathrm{diam}(\vec{r})) \sim p^{-1}k^{(\vec{r})}(0)(sd_0)^{-|\vec{r}|}(s-\mathrm{diam}(\vec{r})) \quad \vec{r}\neq 0\,, \tag{S52}$$

and $\hat{k}(0)$ when $\vec{r} = 0$.

Note that in the case when the stride is the same as the size of the patch, the $(s - \mathrm{diam}(\vec{r}))$ becomes 1 for all spherical harmonics. As such, smaller strides favor functions with smaller diameters (namely, $\mathrm{diam}(\vec{r})$), breaking the symmetry between functions with small and large diameters.

We turn to compute the dimension of r-eigenspace for $r \in \mathbb{N}$. Clearly, for $\vec{r} = 0$ the dimension is 1 and for $|\vec{r}| = 1$ the dimension is $d = pd_0$, which is the dimension of all degree 1 homogenous polynomials. For $|\vec{r}| > 1$, we count the number of spherical harmonics in the 1st patch $\xi_{0:s}$ with $r_0 \neq 0$ and the total number of spherical harmonics in all patches is p time this number. Thus

$$\begin{aligned}
\dim(\mathsf{H}^{(r)}) &= p \sum_{\substack{\vec{r}\in\mathbb{N}^s:\\ |\vec{r}|=r, r_0\neq 0}} N(d_0,\vec{r}) && \text{(S53)}\\
&= p\left(\sum_{\substack{\vec{r}\in\mathbb{N}^s:\\ |\vec{r}|=r}} N(d_0,\vec{r}) - \sum_{\substack{\vec{r}\in\mathbb{N}^s:\\ |\vec{r}|=r, r_0=0}} N(d_0,\vec{r})\right) && \text{(S54)}\\
&\sim \left(\sum_{\substack{\vec{r}\in\mathbb{N}^s:\\ |\vec{r}|=r}} d_0^r/\vec{r}! - \sum_{\substack{\vec{r}\in\mathbb{N}^{s-1}:\\ |\vec{r}|=r}} d_0^r/\vec{r}!\right) && \text{(S55)}\\
&= d_0^r/r!(s^r-(s-1)^r) \sim s^{r-1}d_0^r/(r-1)! && \text{(S56)}
\end{aligned}$$

for large s.

+Locality + Pooling GAP_∞**.** The kernel is given by

$$\mathcal{K}_{\mathsf{GAP}}(\xi,\eta) = \frac{1}{p^2}\sum_{i,j\in[p]} k\left(\frac{1}{s}\sum_{b\in[s]}\xi_{i+b}^T\eta_{j+b}/d_0\right).$$

In what follows we identify $B(\mathbb{N}^p, s)/\tau = B(\mathbb{N}^s, s)$. Applying Theorem D.1 gives

$$\begin{aligned}
\mathcal{K}_{\mathsf{GAP}}(\xi,\eta) &= \frac{1}{p^2}\sum_{i,j\in[p]} k\left(\frac{1}{s}\sum_{b\in[s]}\xi_{i+b}^T\eta_{j+b}/d_0\right),\\
&= \sum_{\vec{r}\in\mathbb{N}^s}\hat{K}(\vec{r})N(d_0,\vec{r})^{-1}\sum_{\vec{l}}\frac{1}{p^2}\sum_{i,j\in[p]} Y_{\vec{r},\vec{l}}(\xi_{i:i+s}/\sqrt{d_0})Y_{\vec{r},\vec{l}}(\eta_{j:j+s}/\sqrt{d_0})\\
&= \hat{K}(0)N(d_0,\vec{0}) + \sum_{\vec{r}\in B(\mathbb{N}^p,s)/\tau, \vec{r}\neq 0}\hat{K}(\vec{r})N(d_0,\vec{r})^{-1}\frac{1}{p}\sum_{\vec{l}} Y^{\tau}_{\vec{r},\vec{l}}(\xi/\sqrt{d_0})Y^{\tau}_{\vec{r},\vec{l}}(\eta/\sqrt{d_0})
\end{aligned}$$

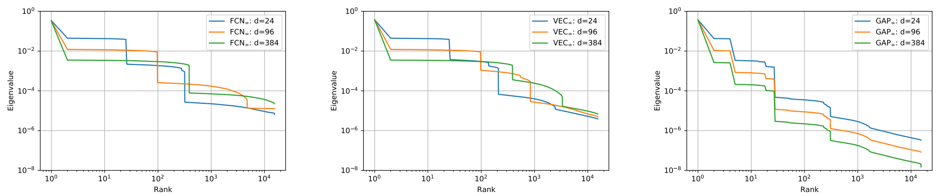

Figure S1: **Eigenvalue Decay of Relu NTK of** FCN_∞**,** VEC_∞ **and** GAP_∞**.** $d_0 = s = 3$. The eigenvalues of GAP_∞ decays *faster* because with $m = 15k$ many samples, higher order eigenspace can be covered by GAP_∞ but not by $\mathsf{FCN}_\infty/\mathsf{VEC}_\infty$ as pionted out in Theorem 4.1.

where we have defined for $\vec{r} \in B(\mathbb{N}^p, s)/\tau$ with $\vec{r} \neq 0$

$$Y^{\tau}_{\vec{r},\vec{l}}(\xi/\sqrt{d_0}) = \frac{1}{\sqrt{p}} \sum_{i\in[p]} Y_{\vec{r},\vec{l}}(\xi_{i:i+s}/\sqrt{d_0}) \tag{S57}$$

The eigvenvalue for $\vec{r} = 0$ is $\hat{k}(0)$ and for $\vec{r} \neq 0$ with $k^{(\vec{r})}(0) \neq 0$ are

$$\hat{K}(\vec{r})N(d_0, \vec{r})^{-1}\frac{1}{p} \sim p^{-1}(sd_0)^{-|\vec{r}|}k^{(|\vec{r}|)}(0) \tag{S58}$$

Similar to VEC_∞, the dimension of r-eigenspace is $s^{r-1}(d_0)^r/(r-1)!$ for $r \geq 1$. $\square$

D.4 Remarks for Theorem D.2

The **Baseline** $\mathcal{K}_{\mathsf{FCN}_n}$ is a standard result; see, for example, [17] and [56]. The dimension of r-degree harmonic polynomials is $\Theta(d^r)$ and the spectral gap between the 0- and r-eigenspaces, namely, the r-condition number, $\kappa_r = \Theta(d^r)$. Learning higher order terms (using kernels) in this space suffers from the curse of dimensionality because (1) the number of samples requires to cover a basis of the r-eigenspace and (2) the number of gradient steps (or the amount of time for gradient flow) needed to learn the r-eigenspace grow with the rate $\Theta(d^r)$. This makes it difficult to learn higher order terms even when d is not very large, e.g., when $r = 4$ and $d = 784$ (Mnist), $d^r \sim 10^{11}$ and lower order terms when d is large, e.g. when $r = 2$ and $d = 3 \times 224^2 \sim 10^5$ (ImageNet), $d^r \sim 10^{10}$.

The **+Locality** $\mathcal{K}_{\mathsf{VEC}}$ dramatically reduces both the dimension of the function space and the spectral gap: $\kappa_r \sim \dim(\mathsf{H}^{(r)}) \sim d(sd_0)^{r-1}$. For example, the first layer of ResNet (applied to ImageNet) is a $(7, 7)$ convolution with stride $(2, 2)$ which corresponds to $sd_0 = 7^2 \times 3 \sim d^{0.42}$, where $0.42 \sim \log(7^2 \times 3)/\log(224^2 \times 3)$. With $m \sim d^r$ samples, $\mathcal{K}_{\mathsf{FCN}}$ could cover the r-eigenspace, while K_{VEC} could cover $1 + (r-1)/0.42 \sim (2.4r - 1.4)$-eigenspace.

The **+Locality+Pooling** $\mathcal{K}_{\mathsf{GAP}}$. The dimension of the function space is reduced by a factor of p to $\dim(\mathsf{H}^{(r)}) \sim (sd_0)^{r-1}d_0$ and the spectral gap $\kappa_r \sim d(sd_0)^{r-1}$ is unchanged. As a result, $\mathcal{K}_{\mathsf{GAP}}$ is p-times more sample-efficient than $\mathcal{K}_{\mathsf{VEC}}$

In all cases above, the r-condition number κ_r can be improved by a factor of d by removing the 0-th eigenspace of the kernels.

D.5 Proof of Theorem D.1

We only need to compute the "Fourier coefficients" $\hat{K}(\vec{r})$. First,

$$\langle P_{\vec{r}}, P_{\vec{r}}\rangle_{L^2(I^p, \omega^p_{d_0})} = \prod_{j\in[p]} \langle P_{r_j}, P_{r_j}\rangle_{L^2(I, \omega_{d_0})} = N(d_0, \vec{r})^{-1}\left(\frac{|\mathbb{S}_{d_0-1}|}{|\mathbb{S}_{d_0-2}|}\right)^p \tag{S59}$$

The last equality could be obtained by applying the addition theorem Lemma 2 and then integrate over $\mathbb{S}^p_{d_0-1}$; see Eq. (4.30) in [55].

To handle the numerator in Equation S30, we assume K is sufficiently smooth to avoid the boundary effect. When this is not the case, a little bit effort is needed to handle the boundary values which will be skipped here. By applying Lemma D.1, integration by parts and continuity of $K^{(\vec{r})}$ on the boundary ∂I^p

$$\langle K, P_{\vec{r}}\rangle_{L^2(I^p,\omega_{d_0}^p)} = c_{\vec{r}} \int_{I^p} K(t) \left(\frac{d}{d\vec{t}}\right)^{\vec{r}} \left(1-\vec{t}^2\right)^{\vec{r}+(d_0-3)/2} d\vec{t} \tag{S60}$$

$$= (-1)^{\vec{r}} c_{\vec{r}} \int_{I^p} K^{(\vec{r})}(t) \left(1-\vec{t}^2\right)^{\vec{r}+(d_0-3)/2} d\vec{t} \tag{S61}$$

$$= (-1)^{\vec{r}} c_{\vec{r}} (\mathcal{M}(K,d_0) + \epsilon(K,d_0)) \tag{S62}$$

where $K^{(\vec{r})}$ is the $\vec{r}$ derivative of K, the coefficient is given by Lemma D.1 and

$$c_{\vec{r}} = \prod_{j\in[p]} c_{r_j} = \prod_{j\in[p]} \frac{(-1)^{r_j}}{2^{r_j}(r_j+(d_0-3)/2)_{r_j}} \sim \prod_{j\in[p]} (-1)^{r_j} d_0^{-r_j} = (-1)^{\vec{r}} d_0^{-\vec{r}} \tag{S63}$$

and the major and error terms are given by

$$\mathcal{M}(K,d_0) = K^{(\vec{r})}(0) \int_{I^p} \left(1-\vec{t}^2\right)^{\vec{r}+(d_0-3)/2} d\vec{t} = K^{(\vec{r})}(0) \prod_{j\in[p]} \frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|} \tag{S64}$$

$$\epsilon(K,d_0) = \int_{I^p} (K^{(\vec{r})}(t) - K^{(\vec{r})}(0)) \left(1-\vec{t}^2\right)^{\vec{r}+(d_0-3)/2} d\vec{t} \tag{S65}$$

For the error term, we use the mean value theorem to bound

$$|(K^{(\vec{r})}(t) - K^{(\vec{r})}(0))| \le \|K\|_{C^{|\vec{r}|+1}(I^p)} \sum_{j\in[p]} |t_j| \tag{S66}$$

and

$$|\epsilon(K,d_0)| \le \|K\|_{C^{|\vec{r}|+1}(I^p)} \int_{I^p} \left(1-\vec{t}^2\right)^{\vec{r}+(d_0-3)/2} d\vec{t} \sum_{j\in[p]} \left(\frac{\int_I |t_j| \left(1-t_j^2\right)^{r_j+(d_0-3)/2} dt_j}{\int_I \left(1-t_j^2\right)^{r_j+(d_0-3)/2} dt_j} \right) \tag{S67}$$

$$\sim \|K\|_{C^{|\vec{r}|+1}(I^p)} \prod_{j\in[p]} \frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|} \sum_{j\in[p]} d_0^{-1} \left(\frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|}\right)^{-1}. \tag{S68}$$

Since for any $\alpha \in \mathbb{N}$, as $d_0 \to \infty$,

$$\frac{|\mathbb{S}_{\alpha+d_0-1}|}{|\mathbb{S}_{\alpha+d_0-2}|} = \pi^{\frac{1}{2}} \Gamma((\alpha+d_0-1)/2)/\Gamma((\alpha+d_0)/2) \sim \pi^{\frac{1}{2}} (d_0/2)^{-\frac{1}{2}} \tag{S69}$$

We have

$$|\epsilon(K,d_0)| \lesssim \|K\|_{C^{|\vec{r}|+1}(I^p)} p d_0^{-\frac{1}{2}} \prod_{j\in[p]} \frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|} \tag{S70}$$

Therefore

$$\langle K, P_{\vec{r}}\rangle_{L^2(I^p,\omega_{d_0}^p)} = c_{\vec{r}} \left(K^{(\vec{r})}(0) + \mathcal{O}(\|K\|_{C^{|\vec{r}|+1}(I^p)} p d_0^{-\frac{1}{2}})\right) \prod_{j\in[p]} \frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|} \tag{S71}$$

Plugging back to Equation S30, we have

$$\hat{K}(\vec{r}) = (-1)^{\vec{r}} c_{\vec{r}} N(d_0,\vec{r}) \left(K^{(\vec{r})}(0) + \mathcal{O}(\|K\|_{C^{|\vec{r}|+1}(I^p)} p d_0^{-\frac{1}{2}})\right) \left(\prod_{j\in[p]} \frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|}\right) \left(\frac{|\mathbb{S}_{d_0-1}|}{|\mathbb{S}_{d_0-2}|}\right)^{-p} \tag{S72}$$

Since, for $\vec{r}$ and as $d_0 \to \infty$

$$\frac{c_{\vec{r}}}{(-1)^{\vec{r}} d_0^{-\vec{r}}} \to 1 \quad \text{and} \quad \frac{N(d_0,\vec{r})}{d_0^{|\vec{r}|}/r!} \to 1 \quad \text{and} \quad \left(\prod_{j\in[p]} \frac{|\mathbb{S}_{2r_j+d_0-1}|}{|\mathbb{S}_{2r_j+d_0-2}|}\right) \left(\frac{|\mathbb{S}_{d_0-1}|}{|\mathbb{S}_{d_0-2}|}\right)^{-p} \to 1 \tag{S73}$$

and thus

$$\hat{K}(\vec{r}) = \vec{r}!^{-1} \left(K^{(\vec{r})}(0) + \mathcal{O}(\|K\|_{C^{|\vec{r}|+1}(I^p)} p d_0^{-\frac{1}{2}})\right) \tag{S74}$$

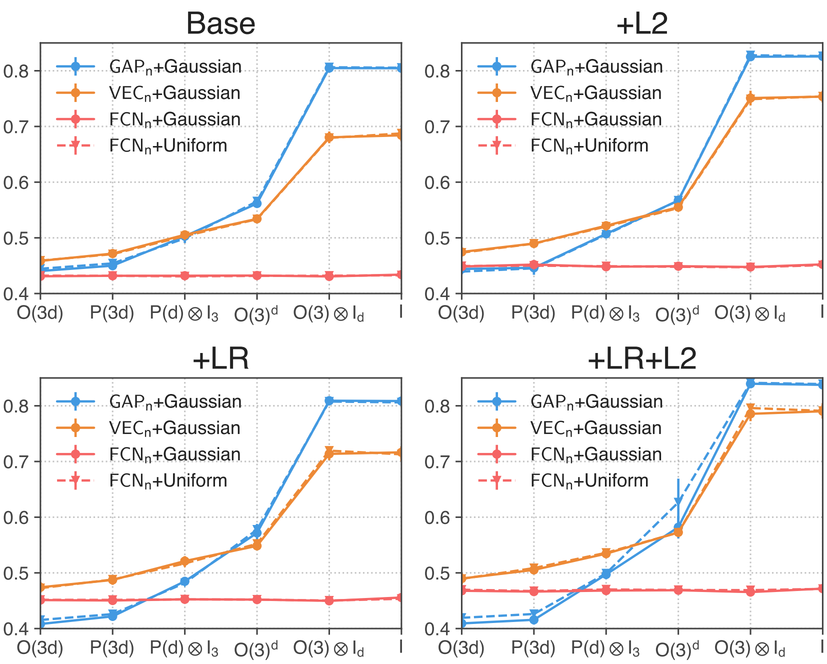

Figure S2: **Replacing the Gaussian initialization by uniform distribution does not change the performance much.**

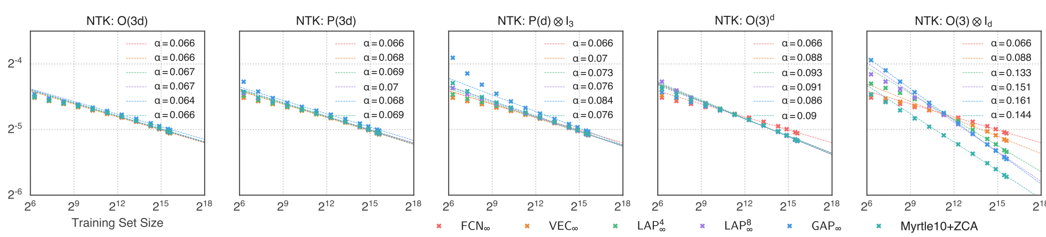

Figure S3: **Scaling Law of Infinite Network vs Different Symmetries.**

E Plots Dump

E.1 Gaussian vs Uniform Initialization

E.2 Scaling Law for Infinite Networks

E.3 Finite Width Effect of VEC_n.

E.4 Implication of Theorem D.2

We investigate the data-efficiency of various architectures on various tasks. The tasks are to learn harmonic polynomials containing degree $r = 1, 24$ in $(\mathbb{S}_2)^{16}$. The MSE of each degree is normalized to be 0.5 and the MSE of the zero predictor is 1.5. There are 5 types of polynomials/tasks (columns in Fig.S5):

1. **Non-local**, which is our baseline, corresponding to generic polynomials without structure information. The optimal kernel to solve this task in this paper is $\mathcal{K}_{\mathsf{FCN}}$.
2. **Non-local+shift**, adding shifting invariance to **Non-local**. The optimal kernel is $\mathcal{K}_{\mathsf{FCN}}$ + Shifting invariance.

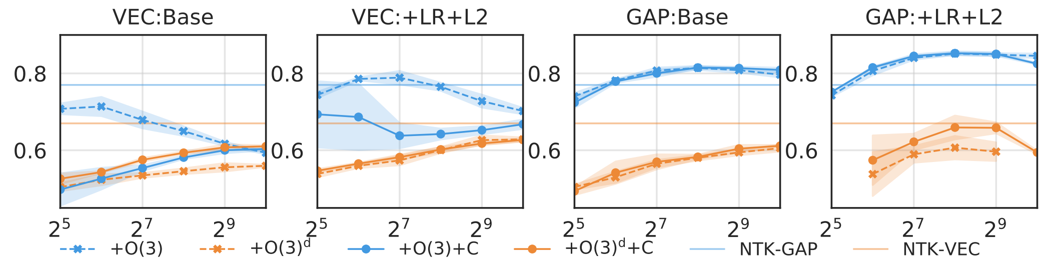

Figure S4: **Performance vs Width for** VEC_n **and** GAP_n With the $\mathrm{O}(3)^d$ symmetry imposed on the system, performance of VEC_n is below the performance of VEC_∞ (67%), but monotonically improves as the width n increases. However, with the original coordinate system ($\mathrm{O}(3) \otimes \mathbf{I}_d$), performance (without centering) improves and then degrades significantly after the peak. This is because the network is less sensitive to the $O(3)$ symmetry. In stark contrast, the performance of GAP_n improves from $n = 32$ to $n = 512$ but only slightly degrades at $n = 1024$. With and without centering, the performance of GAP_n is similar while the performance of VEC is dramatically different.

3. **Local**: the polynomial depends locally on patches of size (3, 3), i.e. $\mathbb{S}_2^3$; The optimal kernel is $\mathcal{K}_{\mathsf{VEC}} = \mathcal{K}_{\mathsf{FCN}}$.

4. **Local + Sparse**: the polynomial depends only on one *single* patch. The optimal kernel should be a FCN-kernel defined on that patch, which is a not available among our kernels. The $\mathcal{K}_{\mathsf{VEC}} = \mathcal{K}_{\mathsf{FCN}}$ is the second best.

5. **Local + Shift**: enforcing shifting invariance **Local**. The optimal kernel is $\mathcal{K}_{\mathsf{GAP}}$.

In the (5, 5)-panel Fig.S5, we plot the MSE (y-axis) vs $\log(m)/\log(d)$ (x-axis), where m is the number of samples and $d = 3 * 16 = 48$ is the dimension of the input data, for different learning algorithms: (1) NNGP, the Gaussian Process kernel (2) NTK, the kernel of infinite width network corresponding to training only the first layer (3) NN, finite width networks with width $n = 16$, (4) $n = 4096$ and (5) n=best, which is obtained as follows: for each m, we sweep over $n = 16 \to 4096$ dyadically by a factor of 2 and report the best performance.

For Non-FCN kernels, we choose m up to $5120 \times 4 \sim 20k$, since the MSE have already reached a very small number, i.e. learning all frequencies $r = 1, 2, 4$. For FCN, we choose m up to $5120 \times 32 \sim 160k$, the biggest $m \times m$ matrix that we could be solved within our compute budget. However this still falls in short with $d^4 = (48)^4 \sim 5000k = 5 \times 10^6$, the dimension of 4-eigenspace. Not surprising, the vanilla FCN kernel could not learn the $r = 4$ frequency for all tasks (first row). However, FCN kernel + Shifting could learn **Non-local+shift** and **Local + Shift** with $m \sim d^3$, since the symmetry *shifting* reduces the dimension of r-eigenspace by a factor of d.

Finite width FCN_n does better than kernels when learning (higher) $r = 4$ frequency, requiring $m \sim d^{3+}$ many samples (first row of the plot), while kernel would require d^4 many samples. It does even better on the task **Local + Sparse** with smaller n and equally less good in **Non-local, Non-local + shift, Local** and **Local + Shift**. This says finite width networks are good at handling *sparsity* but not *locality*, which has to be imposed by human into them as a form of inductive biases.

Now let us focus on the third row LCN_n. Not surprising, it does bad on the first two tasks **Non-local** and $\mathbf{Non-local+shift}$ because the function space is to small. For the remaining tasks, kernels and finite width networks are efficient and competitively with each other. Only in the task $\mathbf{Local + Sparse}$ LCN_n does noticeable better than kernel, demonstrating the strong ability of finite width networks in handling sparsity.

With weight-sharing (4th-row), VEC does noticeably better in all tasks that require locality. It is an interesting direction to understand the analytic reason behind it.

With the correct prior, the GAP_∞ does equally well as GAP_n. Both of them are the most data-efficient among all other architectures/algorithms in the plot when handling the task $\mathbf{Local + Shift}$.

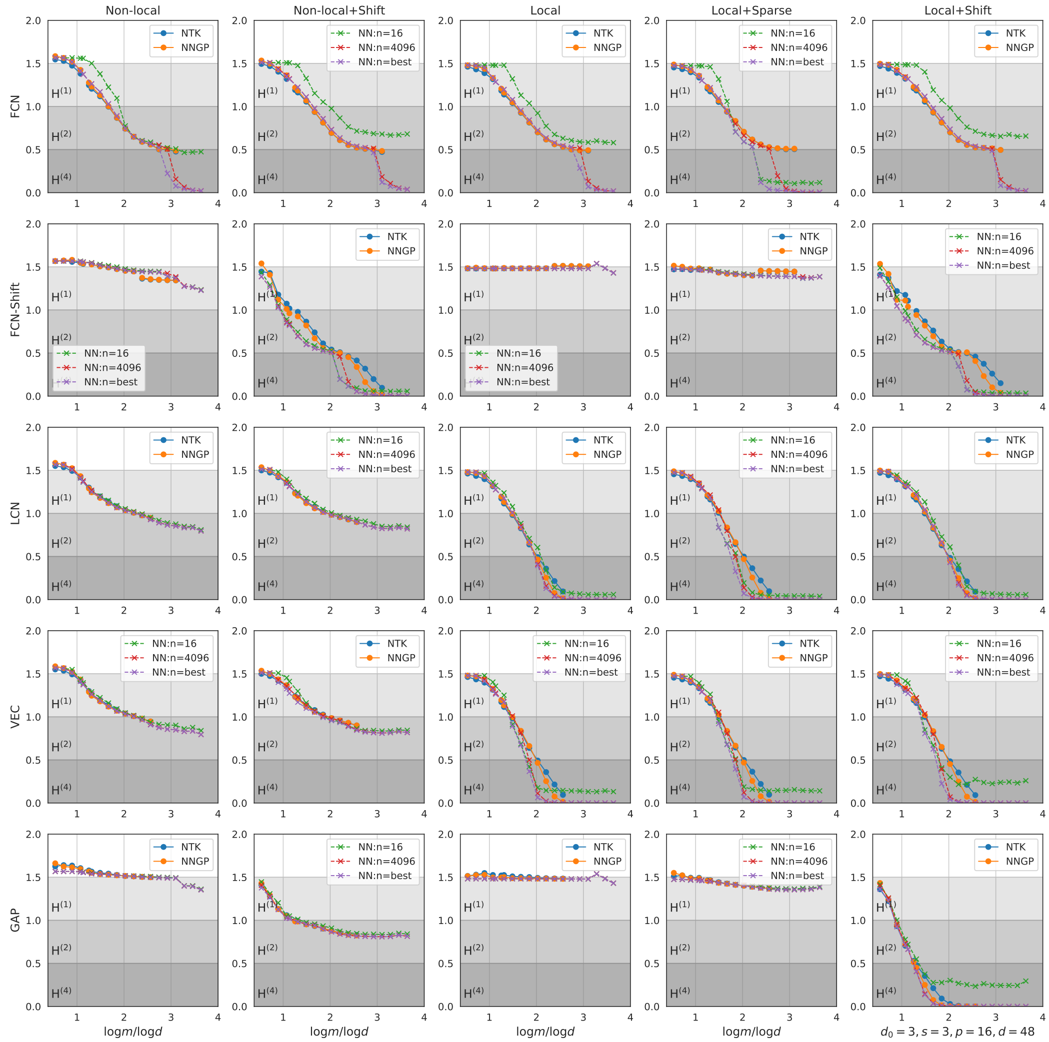

Figure S5: **Impact of Locality and Symmetries.** Performance of 5 types of kernels and finite width networks on 5 types of tasks.

E.5 Scaling Plots for ResNet34 and ResNet101

E.6 ImageNet Samples

F An example for invariance

Example 1. ***Linear Regression:*** *Let $\mathcal{A}$ be the (determnistic) algorithm that outputs the minimum norm linear regrssion solution, then $\mathcal{A}$ is $O(3d)$-invarant, because for $\mathcal{D}_T = (\mathcal{X}_T, \mathcal{Y}_T)$, the prediction*

$$\mathcal{A}(\mathcal{D}_T)(\mathcal{X}_*) \equiv \mathcal{X}_*\mathcal{X}_T^T(\mathcal{X}_T\mathcal{X}_T^T)^\dagger\mathcal{Y}_T = \tau\mathcal{X}_*(\tau\mathcal{X}_T)^T((\tau\mathcal{X}_T)(\tau\mathcal{X}_T)^T)^\dagger\mathcal{Y} \equiv \mathcal{A}^\tau(\mathcal{D}_T)(\mathcal{X}_*), \quad \text{(S75)}$$

where $\tau x \equiv xU_\tau$, here x is a row vector and $U_\tau \in O(3d)$ is the matrix representation of τ.

If $\mathcal{A}$ is the (stochastic) algorithm that applies gradient flow to solve the linear regression $\mathcal{X}_T\omega = \mathcal{Y}_T$ with the MSE loss and the entries of ω are initialized with iid standard Gaussian, then each $f \in \mathcal{A}(\mathcal{D}_T)$ is a draw from the posterior, namely,

$$f(\mathcal{X}_*) \sim \mathcal{N}\left(\mathcal{X}_*\mathcal{X}_T^T(\mathcal{X}_T\mathcal{X}_T^T)^\dagger\mathcal{Y}, \mathcal{X}_*\mathcal{X}_*^T - \mathcal{X}_T\mathcal{X}_T^T(\mathcal{X}_T\mathcal{X}_T^T)^\dagger\mathcal{X}_T^T\mathcal{X}_*\right). \quad \text{(S76)}$$

Note that the distribution is invariant to coordinate rotation by any $\tau \in O(3d)$ and therefore $(\tau\mathcal{D}, \mathcal{M}, \mathcal{I}) = (\mathcal{D}, \mathcal{M}, \mathcal{I})$ for all $\tau \in O(3d)$.

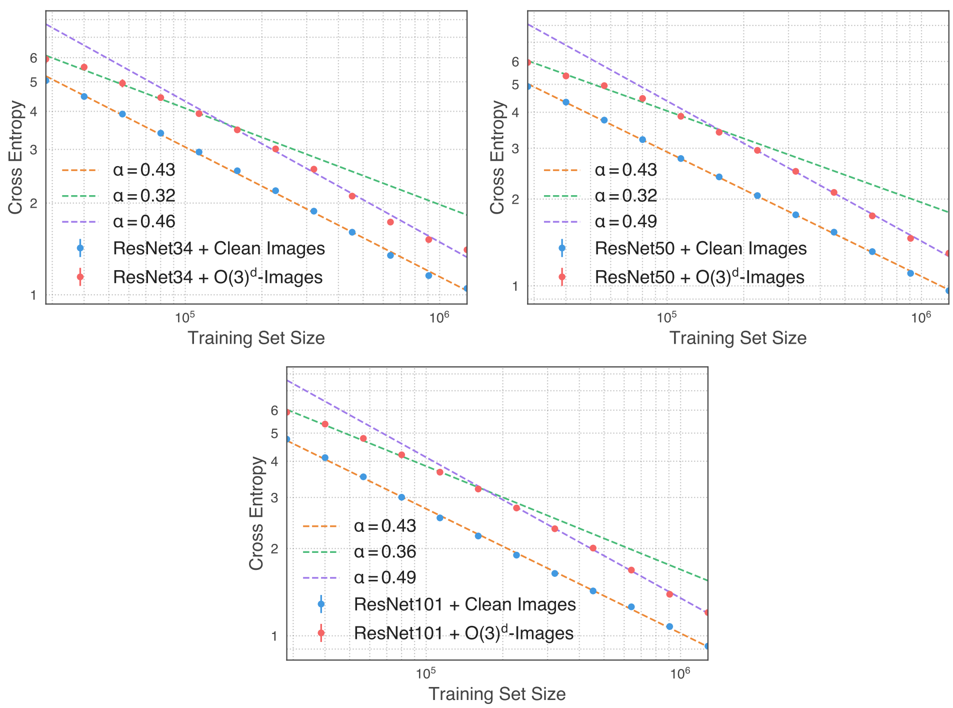

Figure S6: **Scaling vs Rotation**

Figure S7: $\mathrm{O}(3)^d-$**Rotated ImageNet Samples. Seed=1**

Figure S8: $\mathrm{O}(3)^d-$**Rotated ImageNet Samples. Seed=2**

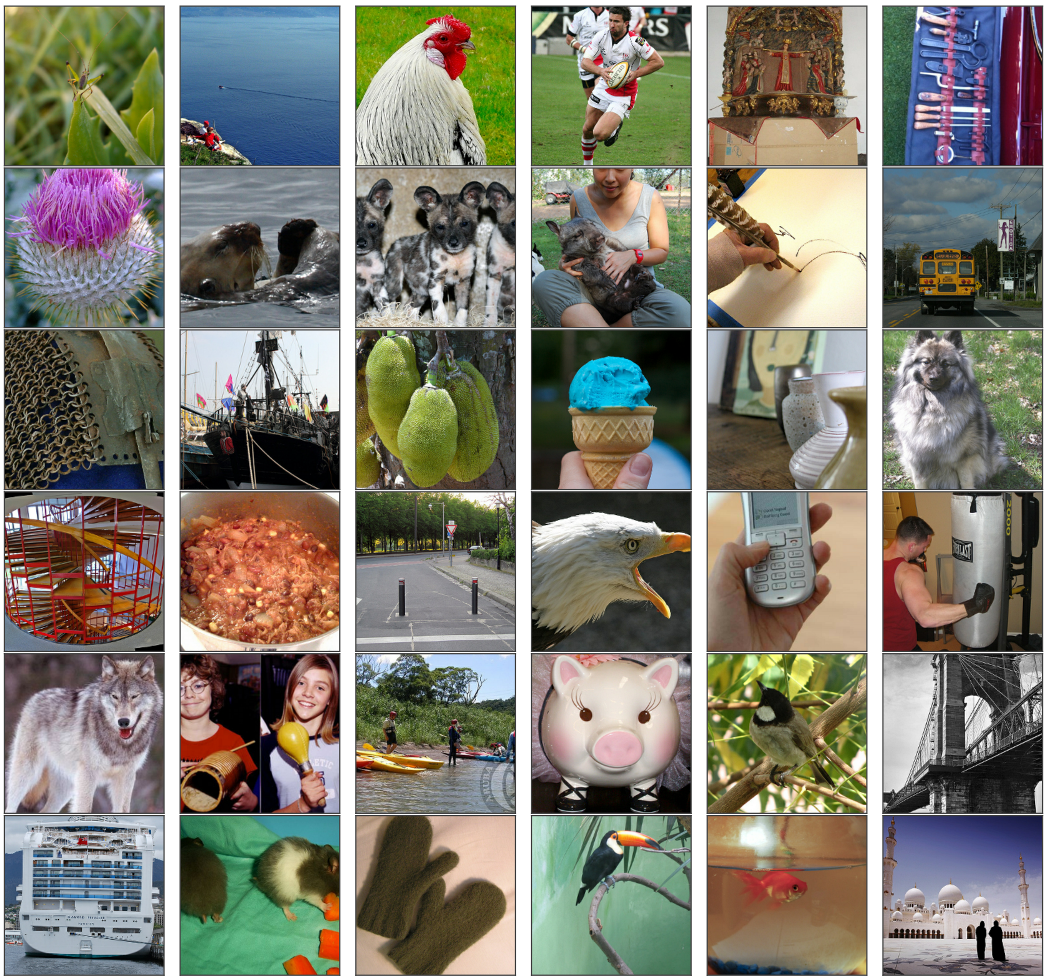

Figure S9: **Clean ImageNet Samples**