Proceedings of Machine Learning Research vol 178:1–55, 2022 35th Annual Conference on Learning Theory

Eigenspace Restructuring: A Principle of Space and Frequency in Neural Networks

Lechao Xiao XLC@GOOGLE.COM
Google Research, Brain Team

Editors: Po-Ling Loh and Maxim Raginsky

Abstract

Understanding the fundamental principles behind the massive success of neural networks is one of the most important open questions in deep learning. However, due to the highly complex nature of the problem, progress has been relatively slow. In this note, through the lens of infinite-width networks, a.k.a. neural kernels, we present one such principle resulting from hierarchical localities. It is well-known that the eigenstructure of infinite-width multilayer perceptrons (MLPs) depends solely on the concept *frequency*, which measures the order of interactions. We show that the topologies from deep convolutional networks (CNNs) restructure the associated eigenspaces into finer subspaces. In addition to frequency, the new structure also depends on the concept *space*, which measures the spatial distance between nonlinear interaction terms. The resulting fine-grained eigenstructure dramatically improves the network's learnability, empowering them to simultaneously model a much richer class of interactions, including Long-Range-Low-Frequency interactions, Short-Range-High-Frequency interactions, and various interpolations and extrapolations in-between. Additionally, model scaling can improve the resolutions of interpolations and extrapolations and, therefore, the network's learnability. Finally, we prove a sharp characterization of the generalization error for infinite-width CNNs (aka C-NTK and CNN-GP) of any depth in the high dimensions. Two corollaries follow: (1) infinite-width deep CNNs can overcome the curse of dimensionality, and (2) scaling improves performance in both the finite and infinite data regimes.

Keywords: NTK, generalization, overparameterized networks, convolutional neural networks

1. Introduction

Learning in high dimensions is commonly believed to suffer from the curse of dimensionality, in which the number of samples required to solve the problem grows rapidly (often polynomially) with the dimensionality of the input (Bishop, 2006). Nevertheless, modern neural networks often exhibit an astonishing power to tackle a wide range of highly complex and high-dimensional real-world problems, many of which were thought to be out-of-scope of known methods (Krizhevsky et al., 2012; Vaswani et al., 2017; Devlin et al., 2018; Silver et al., 2016; Senior et al., 2020; Kaplan et al., 2020). What are the mathematical principles that govern the astonishing power of neural networks? This question perhaps is the most crucial research question in the theory of deep learning because such principles are also the keys to resolving fundamental questions in the practice of machine learning, such as (out-of-distribution) generalization (Zhang et al., 2021), calibration (Ovadia et al., 2019), interpretability (Montavon et al., 2018), robustness (Goodfellow et al., 2014).

Inarguably, there can be more than one of such principles. They are related to one or more of the three basic ingredients of machine learning methods: the data, the model, and the inference algorithm. Among them, the models, a.k.a. architectures of neural networks, are the most crucial innovation in deep learning that set it apart from classical machine learning methods. More importantly,

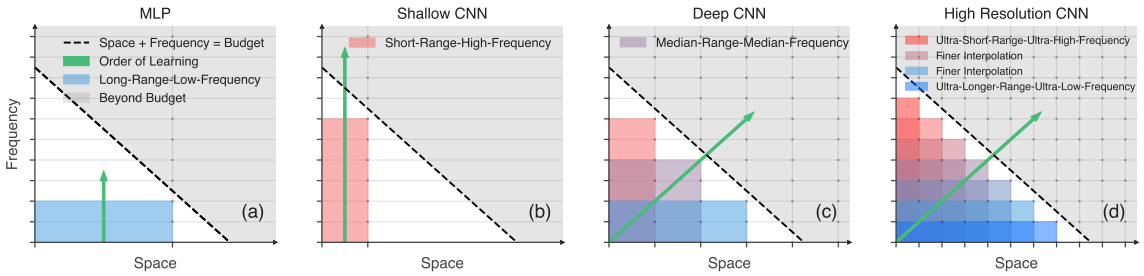


Figure 1: **Architectural Inductive Biases.** A conceptual illustration of learnable functions vs architectures for four families of architectures. Each colored box indicates a learnable eigenspace within a given compute/data budget (**Dashed Line.**) We use **blue**/**red** to indicate *range*/*frequency*. The complexity of an eigenspace is the sum of the complexities in space (X-axis) and in frequency (Y-axis). From left to right, under the same budget: (a) MLPs can model **Long-Range-Low-Frequency (LRLF)** interactions; (b) S-CNNs can model **Short-Range-High-Frequency (SRHF)** interactions; (c) D-CNNs can model not only **LRLF** and **SRHF**, but also various interactions in-between, e.g., **Median-Range-Median-Frequency** interactions. (d) Finally, HS-CNNs can additionally model interactions of **Ultra-Short-Range-Ultra-High-Frequency**, **Ultra-Long-Range-Ultra-Low-Frequency**, and **finer interpolations** in-between. The **Green Arrow** indicates the direction of expansion of learnable eigenspaces when increasing the compute/data budget. See Sec. 6 for more concrete explanations.

the current revolution in machine learning is initialized by the (re-)introduction of convolution-based architectures (Krizhevsky et al., 2012; Lecun, 1989), and subsequent breakthroughs are often driven by discoveries or applications of novel architectures (Vaswani et al. (2017); Devlin et al. (2018)). As such, identifying and understanding the fundamental roles of architectures are of great importance, which is the main focus of the current paper.

In this paper, we take a step forwards by leveraging recent developments in overparameterized networks (Poole et al. (2016); Daniely et al. (2016); Schoenholz et al. (2017); Lee et al. (2018); Matthews et al. (2018b); Xiao et al. (2018); Jacot et al. (2018); Du et al. (2018); Novak et al. (2019a); Lee et al. (2019); Yang (2019), among many others.) These developments have discovered an important connection between neural networks and kernel machines: the Neural Network Gaussian Process (NNGP) kernels and the neural tangent kernels (NTKs). Under certain scaling limits, the former describes the distribution of the outputs of a randomly initialized network (a.k.a. *prior*), and the latter can describe the network's gradient descent dynamics. Although recent work (Ghorbani et al., 2019; Yang and Hu, 2020; Chizat et al., 2019) has identified several limitations of using them in studying the feature learning dynamics of practical networks, we show that they do capture several crucial and perhaps surprising properties of the architectural inductive biases.

Our main contribution is the *eigenspace restructuring* theorem. When the input dimension is sufficiently large and the widths of the network approach infinity, it characterizes a mathematical relation between the network's architecture and its learnability through a trade-off between *space* (spatial complexity) and *frequency* (frequency complexity), providing novel insights behind the mystery power of deep CNNs (more generally, hierarchical localities (Deza et al., 2020; Vasilescu et al., 2021).) We summarize our main contributions below; see Fig. 1 for a conceptual illustration.

1. We introduce a new complexity measure called *learning index* (LI) that precisely characterizes the learnability, generalization and the order of learning of infinite-width networks in high dimensions.

2. We establish the mathematical relation between the network's topology and its learning index.

3. We show that deep CNNs can overcome the curse of dimensionality. Moreover, deeper CNNs generalize better (a.k.a. generalization depth separation.)

4. Finally, we verify our theoretical claims empirically for both neural kernels and finite-width networks (trained by SGD + Momentum) for datasets and networks of practical sizes (Sec.C.)

Overall, our results show that the benefits from architectural inductive biases can be huge even in the infinite-width (kernel) setting. The rest of the paper is organized as follows. Sec. 2 provides a toy example to help understand the motivations of the paper and to explain two core concepts: *spatial* and *frequency* complexities, which jointly define a function-and-architecture dependent complexity measure. Sec. 3 briefly recaps the role of eigenstructures in studying the learning dynamics of linear models, and the connection between (in)finite-width networks and kernels (NNGP kernels and NTKs). Sec. 4 introduces the main notations and expresses the neural network computations via directed acyclic graphs (DAGs). The definition of the learning index and the main results are presented in Sec. 5. We provide interpretations and experimental support for the main results in Sec.6 and Sec.C, resp. Finally, additional related work and further discussions are in Sec. 7 and Sec. 8.

2. Motivation and a Toy Example

Before diving into the technical details, it is helpful to have one toy example in mind. Consider learning the following polynomials in $\mathbb{S}_{d-1} \equiv \{x \in \mathbb{R}^d : \|x\|_2 = 1\}$ using (in)finite-width neural networks and for concreteness we have set $d = 10$:

$$f_1(x) = x_9,\ \ f_2(x) = x_0x_1,\ \ f_3(x) = x_0x_8,\ \ f_4(x) = x_6x_7(x_6^2 - x_7^2),\ \ f_5(x) = x_2x_3x_5 \qquad (1)$$

Which architectures (e.g. MLPs, CNNs) can efficiently learn f_i or the sum of f_i? More precisely, (1) if we have sufficient training data, how much time (compute) is required to learn f_i for a given architecture? (2) Alternatively, if we have a sufficient amount of compute, how much data is needed to learn f_i? To address these questions, one crucial step is to provide a meaningful definition of the "learning complexity" of a function f_i for a given architecture $\mathcal{M}$. Denote this complexity associated to compute and to data by $\mathcal{C}_C(f_i; \mathcal{M})$ and $\mathcal{C}_D(f_i; \mathcal{M})$, resp. With such complexity properly defined, the questions are reduced to solving the min-max problem $\min_{\mathcal{M}} \max_i \{\mathcal{C}_{C/D}(f_i; \mathcal{M})\}$, if the task is, e.g., to learn the sum of f_i.

Let's focus on the complexity. For infinite-width MLPs, a.k.a. inner product kernels, it is well-known that they have the inductive biases (Bach, 2017; Yang and Salman, 2020; Ghorbani et al., 2020; Mei et al., 2021a) (often known as the frequency biases) that the model prioritizes learning low-frequency modes (i.e., low degree polynomials) over high-frequency modes. In addition, the models require $\sim d^r$ many data points to learn *any* degree r polynomials in $\mathbb{S}_{d-1}$. The frequency biases of MLPs are the consequence of the fact that the eigenspaces of inner product kernels are structured based only on *frequencies*. Specific to our example, for MLPs, the order of learning is

$f_1/f_2, f_3/f_5/f_4$ and it requires about $d/d^2, d^2/d^4$ ($d = 10$ in the example here) many data points to learn the functions. Clearly, the model is very inefficient in learning f_4 and, more generally, any high-frequency modes.

To design new models that improve the learning efficiency, we must take the modality of the functions into account, which MLPs have overlooked. We observe that: (1) although of high-frequency, f_4 depends only on two consecutive terms x_6 and x_7, which are *spatially* close; (2) in contrast, $f_3(x) = x_0x_8$ is of low-frequency but the spatial distance between the two interaction terms x_0 and x_8 are conceptually "large"; (3) the function $f_5(x) = x_2x_3x_5$ is somewhere in-between: the order of interaction is 3 (lower than that of f_4) and the spatial distance (not yet defined) among interaction terms is conceptually "smaller" than that of $f_3(x) = x_0x_8$, but "greater" than the terms in f_4. Using the terminologies from the introduction Fig. 1, the functions $f_2/f_3/f_5/f_4$ model interactions of types: Short-Range-Low-Frequency/Long-Range-Low-Frequency/Median-Range-Median-Frequency/Short-Range-High-Frequency. By *Range* we mean the distance among the nonlinear interacting terms, and by *Frequency* we mean the order(=degree) of interactions. Clearly, the MLPs are inefficient since they totally ignore the "spatial structure" of the functions. As such, a good architecture must balance the "spatial structure" and the "frequency structure" of the functions. For the same reason, a good complexity measure (1) must be able to capture both the *frequency* of the functions and the *spatial distance* among interaction terms; (2) must be able to precisely characterize the data and the computation efficiency of learning and their dependence on architectures. The learning index mentioned in the introduction is defined to meet these two criteria. It is the sum of the frequency index and the spatial index. The former measures the order (=degree=frequency) of interactions, depending on how the network partitions the input into patches. The latter measures the spatial distance among the interaction terms, depending on how the network hierarchically organizes these patches. Later we show that, in the high-dimensional setting, the learning index provides a sharp characterization for the learnability (in terms of compute and learning efficiency) of eigenfunctions, and certain CNNs can perfectly balance the learning of f_3 and f_4, i.e., informally

$$\mathcal{C}_{C/D}(f_3; \text{CNN}) \approx \mathcal{C}_{C/D}(f_4; \text{CNN}) \approx \mathcal{C}_{C/D}(f_{2/3}; \text{MLP}) << \mathcal{C}_{C/D}(f_4; \text{MLP}) \tag{2}$$

A crucial step in the paper is to understand the exact meaning of *spatial distance* and define it properly. It turns out that this distance and thus the spatial index relies on the topology of the network through the length of the minimum spanning tree(s) (MSTs) connecting the nonlinear interacting terms of the eigenfunction of interest to the output of the network. We explain, conceptually, why MSTs emerge in the definition of the spatial index below. Heuristically, we need to compute the dependence of the output function (i.e., the function defined by the network) on an eigenfunction of interest, e.g., f_5, which is a polynomial in our setting. This amounts to computing certain mixed derivatives w.r.t. to the interacting terms in f_5, i.e. $\partial_{x_2}\partial_{x_3}\partial_{x_5}$. After applying the chain rule and the product rule, this dependence can be expressed as a sum of "paths" (possibly with duplicated edges) from the output to the interacting terms. This sum is intractable in general. However, in certain scaling limits, one can indeed compute the exponent of the leading terms w.r.t. to the input dimension d. Not surprisingly, this exponent is the infimum of the lengths of all such "paths", namely, the length of the MST(s) connecting interacting terms to the output. To formally define the spatial index, we need to express the neural network computations using DAGs (Sec.4) and properly define the shapes of DAGs in high dimensions (Sec. 5).

3. Linear and linearized models

As a warm-up exercise, we briefly go through the training dynamics of a linear regression model. The goal is to explain the relation between eigenstructures and training dynamics.

3.1. Linear Regression

Let $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R} : i \in [m]\}$ be the training set, where $m \in \mathbb{N}$. For convenience, we also use $\mathcal{X} \in \mathbb{R}^{m\times d}$ and $\mathcal{Y} \in \mathbb{R}^m$ to denote the input matrix and label vector resp., where the i-th row of $\mathcal{X}$ and $\mathcal{Y}$ are $\boldsymbol{x}_i^T$ and y_i resp. Let $J : \mathbb{R}^d \to \mathbb{R}^{1\times n}$ be a feature map and $J(\mathcal{X}) \in \mathbb{R}^{m\times n}$ denote the features of the inputs. The task is to learn a linear function $f(\boldsymbol{x}, \theta) \equiv J(\boldsymbol{x})\theta$ to minimize the MSE of the residual $\mathcal{R}(\mathcal{X}, \theta) \equiv f(\mathcal{X}, \theta) - \mathcal{Y}$,

$$\frac{1}{2}\|\mathcal{R}(\mathcal{X}, \theta)\|_2^2 \equiv \frac{1}{2}\sum_{i\in[m]} (f(\boldsymbol{x}_i, \theta) - y_i)^2 .$$

Here $\theta \in \mathbb{R}^n$, a column vector, is the (trainable) parameter of the linear model. Then, by the chain rule, the gradient flow dynamics can be written as

$$\frac{d}{dt}\mathcal{R}(\mathcal{X}, \theta) = -J(\mathcal{X})J^T(\mathcal{X})\mathcal{R}(\mathcal{X}, \theta) \equiv -\mathcal{K}(\mathcal{X}, \mathcal{X})\mathcal{R}(\mathcal{X}, \theta) . \tag{3}$$

Since the feature kernel $\mathcal{K}(\mathcal{X}, \mathcal{X}) = J(\mathcal{X})J^T(\mathcal{X}) \in \mathbb{R}^m \times \mathbb{R}^m$ is constant in time, the above ODE can be solved in closed form. Let $\hat{\mathcal{K}}(j)/\boldsymbol{u}_j$ be the j-th eigenvalue/eigenvector of $\mathcal{K}(\mathcal{X}, \mathcal{X})$ in descending order. By initializing $\theta = 0$ at time $t = 0$ and denoting the projection by $\eta_j = \boldsymbol{u}_j^T\mathcal{R}(\mathcal{X}, 0)$, the dynamics of the residual and the loss can be reduced to

$$\mathcal{R}(\mathcal{X}, \theta_t) = \sum_{j\in[m]} e^{-\hat{\mathcal{K}}(j)t}\eta_j\boldsymbol{u}_j, \quad \mathcal{L}(\theta_t) = \frac{1}{2}\sum_{j\in[m]} e^{-2\hat{\mathcal{K}}(j)t}\eta_j^2 . \tag{4}$$

Therefore, to make the residual loss in $\boldsymbol{u}_j$ smaller than some $\epsilon > 0$, namely, $\frac{1}{2}(e^{-\hat{\mathcal{K}}(j)t}\eta_j)^2 < \epsilon$, the amount of time needed is $t > \log\frac{2\epsilon}{\eta_j^2}/(2\hat{\mathcal{K}}(j))$. The larger $\hat{\mathcal{K}}(j)$ is, the shorter amount of time it takes to learn $\boldsymbol{u}_j$. In addition, if we know the distribution of $\{\sum_{j>i}\eta_j^2 : i \in [m]\}$, then we can plot the scaling law of the loss (Kaplan et al., 2020; Bahri et al., 2021), which is roughly $(\hat{\mathcal{K}}(i)^{-1}, \sum_{j>i}\eta_j^2)_{i\in[m]}$.

Although simple, linear models provide us with the most useful intuition behind the relation between the eigenstructure of the kernel matrix and the learning dynamics of the associated network.

3.2. Linearized Neural Networks: NNGP Kernels and NT Kernels

Let $f(x, \theta)$ be a general function, e.g. f is a neural network parameterized by θ. Similarly,

$$\frac{d}{dt}\mathcal{R}(\mathcal{X}, \theta) = -J(\mathcal{X}, \theta)J^T(\mathcal{X}, \theta)\mathcal{R}(\mathcal{X}, \theta) \equiv -\mathcal{K}(\mathcal{X}, \mathcal{X}; \theta)\mathcal{R}(\mathcal{X}, \theta) . \tag{5}$$

However, the kernel $\mathcal{K}(\mathcal{X}, \mathcal{X}; \theta)$ depends on θ via the Jacobian $J(\mathcal{X}, \theta)$ of $f(\mathcal{X}, \theta)$, and evolves with time. The above system is unsolvable in general. However, under certain parameterization methods (namely, the NTK-parameterization, see Sohl-Dickstein et al. (2020)) and when the network is

sufficiently wide, this kernel does not change much during training and converges to a deterministic kernel called the NTK (Jacot et al., 2018),

$$\mathcal{K}(\mathcal{X}, \mathcal{X}; \theta) \to \Theta(\mathcal{X}, \mathcal{X}) \quad \text{as width} \to \infty. \tag{6}$$

The residual dynamics becomes a constant coefficient ODE again $\dot{\mathfrak{R}}(\mathcal{X}, \theta) = -\Theta(\mathcal{X}, \mathcal{X})\mathfrak{R}(\mathcal{X}, \theta)$. To solve this system, we need the initial value of $\mathfrak{R}(\mathcal{X}, \theta)$. Since the parameters θ are often initialized with iid standard Gaussian variables, as the width approaches infinity, the logits $f(\mathcal{X}, \theta)$ converge to a Gaussian process (GP), known as the neural network Gaussian process (NNGP). Specifically, $f(\mathcal{X}, \theta) \sim \mathcal{N}(\mathbf{0}; \mathfrak{K}(\mathcal{X}, \mathcal{X}))$, where $\mathfrak{K}$ is the NNGP kernel. Note that one can also treat infinite-width networks as Bayesian models, a.k.a. Bayesian Neural Networks, and apply Bayesian inference to compute the posteriors. This approach is equivalent to training *only* the network's classification layer (Lee et al., 2019) and the gradient descent dynamics is described by the kernel $\mathfrak{K}$. As such, there are two natural kernels, the NTK Θ and the NNGP kernel $\mathfrak{K}$, associated to infinite-width networks, whose training dynamics are governed by constant coefficient ODEs. To make progress, it is tempting to apply Mercer's Theorem to eigendecompose Θ and $\mathfrak{K}$, e.g.,

$$\mathfrak{K}(x, \bar{x}) = \sum \hat{\mathfrak{K}}(j)\phi_j(x)\phi_j(\bar{x}) \quad \text{and} \quad \Theta(x, \bar{x}) = \sum \hat{\Theta}(j)\psi_j(x)\psi_j(\bar{x}) \tag{7}$$

Unfortunately, this decomposition is too coarse to be useful since it does not provide fine-grained information about the eigenstructures. E.g, it is not clear what are the corrections to Eq. (7) when changing the architecture from a 2-layer CNN to a 4-layer CNN. For this reason, we choose to work on "concrete" input spaces (product of hyperspheres) with richer mathematical structures on which we can perform calculus (namely, harmonic analysis on spheres). Our primary goal is to characterize the precise analytical dependence of the decomposition Eq. (7) on the network's topology in the high-dimensional limit.

4. Neural Computations on DAGs

This section aims to express the finite-width and the infinite-width neural network computations through DAGs, which are needed for us to define the spatial and frequency indices.

For a positive integer p, let $\mathbb{S}_{p-1}$ denote the unit sphere in $\mathbb{R}^p$ and $\overline{\mathbb{S}}_{p-1} = \sqrt{p}\mathbb{S}_{p-1}$, the sphere of radius $\sqrt{p}$ in $\mathbb{R}^p$. We introduce the normalized sum (integral)

$$\fint_{x\in X} f(x) \equiv |X|^{-1} \sum_{x\in X} f(x) \quad \left(\quad \fint_{x\in X} f(x) \equiv \mu(X)^{-1} \int_{x\in X} f(x)\mu(dx) \right) \tag{8}$$

where X is a finite set (a measurable set with a finite positive measure μ).

We find it more convenient to express the computations in neural networks, and in neural kernels via DAGs (Daniely et al., 2016), as both computations are of *recursive* nature. The associated DAG of a network can be thought of as the original network by setting all its widths (or the number of channels for CNNs) to 1. As such, changing the widths of a network won't alter the associated DAG. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote a DAG, where $\mathcal{N}$ and $\mathcal{E}$ are the nodes and edges, resp. We always assume the graph to have a unique output node $o_{\mathcal{G}}$ and is an ancestor of all other nodes. Denote $\mathcal{N}_0 \subseteq \mathcal{N}$ the set of input nodes (leaves) of $\mathcal{G}$, i.e., the collection of nodes with no child. Each node

$u \in \mathcal{N}$ is associated with a pointwise function $\phi_u : \mathbb{R} \to \mathbb{R}$, which is normalized in the sense $\mathbb{E}_{z\in\mathcal{N}(0,1)}\phi_u^2(z) = 1$. It induces a function $\phi_u^* : I \equiv [-1, 1] \to I$ defined to be

$$\phi_u^*(t) = \mathbb{E}_{(z_1,z_2)\in\mathcal{N}_t}\phi_u(z_1)\phi_u(z_2)\,.$$

Here $\mathcal{N}_t$ denotes a pair of standard Gaussians with correlation t. We associate each $u \in \mathcal{N}$ a finite-dimensional Hilbert space $\mathbb{H}_u$, and each $uv \in \mathcal{E}$ (where the first node u is the parent) a bounded linear operator $\mathcal{L}_{uv} : \mathbb{H}_v \to \mathbb{H}_u$. Let

$$\boldsymbol{\mathcal{X}} \equiv \prod_{u\in\mathcal{N}_0} \boldsymbol{\mathcal{X}}_u \equiv \prod_{u\in\mathcal{N}_0} \overline{\mathbb{S}}_{\dim(\mathbb{H}_u)-1} \subseteq \prod_{u\in\mathcal{N}_0} \mathbb{H}_u \quad \text{and} \quad \boldsymbol{I} = I^{|\mathcal{N}_0|}$$

be the input *tensors* and the input *correlations* to the graph $\mathcal{G}$, resp. We associate two types of computations to a DAG: finite-width neural network computation and infinite-width kernel computation,

$$\mathcal{N}_\mathcal{G} : \boldsymbol{\mathcal{X}} \to \mathbb{H}_{o_\mathcal{G}} \quad \text{and} \quad \mathcal{K}_\mathcal{G} : \boldsymbol{I} \to I\,, \tag{9}$$

resp. They are defined recursively as follows

$$\mathcal{N}_u(\boldsymbol{x}) = \phi_u\left(\sum_{v:uv\in\mathcal{E}} \mathcal{L}_{uv}(\mathcal{N}_v(\boldsymbol{x}))\right) \qquad \text{if} \quad u \notin \mathcal{N}_0 \quad \text{else} \quad \mathcal{N}_u(\boldsymbol{x}) = \boldsymbol{x}_u \tag{10}$$

$$\mathcal{K}_u(\boldsymbol{t}) = \phi_u^*\left(\fint_{v:uv\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t})\right) \qquad \text{if} \quad u \notin \mathcal{N}_0 \quad \text{else} \quad \mathcal{K}_u(\boldsymbol{t}) = \boldsymbol{t}_u \tag{11}$$

where $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ and $\boldsymbol{t} \in \boldsymbol{I}$. The outputs of the computations are $\mathcal{N}_\mathcal{G}(\boldsymbol{x}) = \mathcal{N}o_\mathcal{G}(\boldsymbol{x})$ and $\mathcal{K}_\mathcal{G}(\boldsymbol{t}) = \mathcal{K}_{o_\mathcal{G}}(\boldsymbol{t})$. Note that $\mathcal{K}_\mathcal{G}$ is indeed the NNGP kernel (Neal, 1996; Daniely et al., 2016; Lee et al., 2018; Matthews et al., 2018a). The NTK (Jacot et al., 2018) can also be written recursively as

$$\Theta_u(\boldsymbol{t}) = \dot{\phi}_u^*\left(\fint_{v:uv\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t})\right) \fint_{v:uv\in\mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \quad \text{with} \quad \Theta_\mathcal{G} = \Theta_{o_\mathcal{G}}\,. \tag{12}$$

Here, $\Theta_u = 0$ if $u \in \mathcal{N}_0$ and $\dot{\phi}_u^*$ is the derivative of ϕ_u^*. See Table 1 in Novak et al. (2020) for more details regarding the computations of NNGP kernels and NTKs. For formal proofs of the convergence of the NNGP kernels and NTKs, see Daniely et al. (2016); Novak et al. (2019a); Yang (2019); Arora et al. (2019).

4.1. Three Examples: MLPs, S-CNNs and D-CNNs.

To unpack the notation in Sec. 4, we consider three concrete examples: an L-hidden layer MLP, a shallow convolutional network (S-CNN) that contains only one convolutional layer and a deep convolutional network (D-CNN) that contains $(1 + L)$ convolutional layers. The architectures are

MLP: $[\textit{Input}] \to [\textit{Dense-Act}]^{\otimes L} \to [\textit{Dense}]$ (13)

S-CNN: $[\textit{Input}] \to [\textit{Conv}(p)\textit{-Act}] \to [\textit{Flatten-Dense}]$ (14)

D-CNN: $[\textit{Input}] \to [\textit{Conv}(p)\textit{-Act}] \to [\textit{Conv}(k)\textit{-Act}]^{\otimes L} \to [\textit{Flatten-Dense-Act}] \to [\textit{Dense}]$ (15)

where p/k is the filter size of the first/hidden layers and $Dense/Conv/Act/Flatten$ means an dense / convolutional / activation / flattening layer. We choose the stride to be the same as the size of the filter for all convolutional layers and choose *flattening* as the readout strategy rather than pooling. The DAGs associated to a 4-layer MLP, a (1 + 1)-layer CNN (with $p = k = d^{\frac{1}{2}}$) and a (1+3)-layer CNN (with $p = k = d^{\frac{1}{4}}$) are a linked list, a depth-2 tree, and a depth-4 tree, resp; see Fig. 2. For more detailed descriptions, see Sec. B in the appendix.

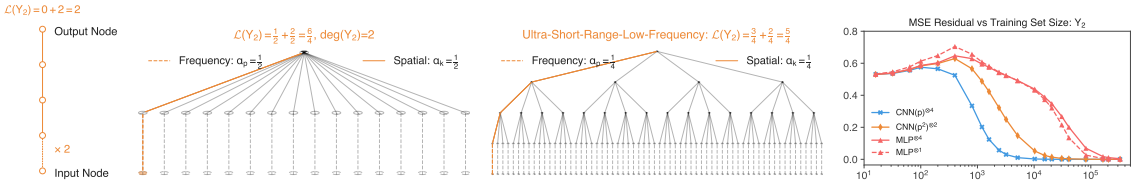


Figure 2: **Architectures/DAGs. vs Eigenfunctions vs Learning Indices.** Left to right: DAGs associated to (a) a four-layer MLP; (b) CNN$(p^2)^{\otimes 2}$, a"D"-CNN that has two convolutional layers (c) CNN$(p)^{\otimes 4}$, a "HR"-CNN that has four convolutional layers; and (d) MSE (Y-axis) vs training set size (X-axis) for $\mathbf{Y_2}$ obtained by NTK-regression for 4 architectures. Here $\mathbf{Y_2}$ is a linear combination of eigenfunctions of Short-Range-Low-Frequency interactions ($\deg(\mathbf{Y_2})=2$); see Sec. D for the expression. The DAGs are generated with $p=4$. In each DAG, the **Dashed Lines** represent the edges with zero weights. The **Solid Lines** have weights 0, $\frac{1}{2}$ and $\frac{1}{4}$ in (a), (b) and (c), resp. The **colored path** represents the minimum spanning tree used to compute the spatial indices of $\mathbf{Y_2}$. Under architectures (a), (b) and (c), the spatial indices are 0, $\frac{1}{2}$ and $\frac{3}{4}$, resp. Each input node represents an input patch of dimension $p^4=d^1$, $p^2=d^{\frac{1}{2}}$ and $p=d^{\frac{1}{4}}$ and the frequency indices are 2×1, $2\times\frac{1}{2}$ and $2\times\frac{1}{4}$ in (a), (b) and (c), resp.

5. Main Results

The goal is to obtain a precise characterization of the relation between the eigenstructures of $\mathcal{K}$ / Θ and the DAG associated to the network's architectures in the large input dimension setting. As such we consider a sequence of graphs $\boldsymbol{\mathcal{G}}=\left(\mathcal{G}^{(d)}\right)_{d\in\mathbb{N}}$, where $\mathcal{G}^{(d)}=(\mathcal{N}^{(d)},\mathcal{E}^{(d)})$. We associate a *finite* set of non-negative numbers $\Lambda_{\boldsymbol{\mathcal{G}}}$ to $\boldsymbol{\mathcal{G}}$, which is called the shape parameters of $\boldsymbol{\mathcal{G}}$,

$$0\in\Lambda_{\boldsymbol{\mathcal{G}}}\subseteq[0,1]\quad\text{and}\quad|\Lambda_{\boldsymbol{\mathcal{G}}}|<\infty. \tag{16}$$

For the rest of the paper, we will use the following notations. For $A,B:\mathbb{N}\to\mathbb{R}^+$,

$$B(d)\gtrsim A(d)\iff A(d)\lesssim B(d)\iff\exists c,\,d_0>0\quad s.t.\quad B(d)\geq cA(d)>0\quad\text{for all}\quad d>d_0$$

and $B(d)\sim A(d)$ if and only if $B(d)\gtrsim A(d)$ and $A(d)\gtrsim B(d)$.

5.1. Assumptions

We need several technical assumptions on $\boldsymbol{\mathcal{G}}$ regarding the asymptotic shapes of $\mathcal{G}^{(d)}$, which are summarized as **Assumption-$\boldsymbol{\mathcal{G}}$** below.

Assumption-$\boldsymbol{\mathcal{G}}$. Let $\boldsymbol{\mathcal{G}}=(\mathcal{G}^{(d)})_d$. There are absolute constants $c,C>0$ and $d_0>0$ such that the followings hold for $d\geq d_0$.

(a.) For each *non-input* node $u\in\mathcal{N}^{(d)}$, there is $\alpha_u\in\Lambda_{\boldsymbol{\mathcal{G}}}$ such that

$$cd^{\alpha_u}\leq\deg(u)\leq Cd^{\alpha_u}. \tag{17}$$

For each edge $uv\in\mathcal{E}^{(d)}$, its weight is defined to be $\pi_{uv}\equiv\alpha_u$.

(b.) For each *input* node v, there are $d_v\in\mathbb{N}$ and $0<\alpha_v\in\Lambda_{\boldsymbol{\mathcal{G}}}$ such that

$$cd^{\alpha_v}\leq d_v\leq Cd^{\alpha_v}\quad\text{and}\quad\sum\nolimits_{v\in\mathcal{N}_0^{(d)}}d_v=d. \tag{18}$$

(c.) Let $\mathcal{N}_1^{(d)} \equiv \{u : \exists v \in \mathcal{N}_0^{(d)} \text{ s.t. } uv \in \mathcal{E}^{(d)}\}$ be the collection of nodes in the *first* hidden layer. We assume that for every $u \in \mathcal{N}_1^{(d)}$, $\alpha_u = 0$ and all children of u are input nodes.

(d.) Every node $v \in \mathcal{N}^{(d)}$ has at most C many parents, i.e. $|\{u : uv \in \mathcal{E}^{(d)}\}| \leq C$. Moreover, the number of *layers* is uniformly bounded, namely, for any node u, any path from u to $o_{\mathcal{G}}$ contains at most C edges.

The first two assumptions also help create spectral gaps between eigenspaces. When d is not large, the "finite-width" effect is no longer negligible, and we expect that the spectra decay more smoothly. Assumption (c.) says there are no (skip) connections from the input layer to other layers except to the first hidden layer, which is often the case in practice.

We say ϕ^* is semi-admissible if, for all $r \geq 1$, the r-th derivative of ϕ^* at zero is non-vanishing, i.e., $\phi^{*(r)}(0) > 0$. If, in addition, $\phi^*(0) = 0$ (i.e., the activation is centered), then we say ϕ is admissible. An activation ϕ is (semi-)admissible if ϕ^* is (semi-)admissible. Note that if ϕ^* is (semi-)admissible, then $\dot{\phi}^*$ is semi-admissible. **Assumption-ϕ.** We make the following assumptions on the activations. (a.) If $u \in \mathcal{N}_0^{(d)}$, ϕ_u is the identity function. (b.) If $u \notin \mathcal{N}_0^{(d)} \cup \{o_{\mathcal{G}}\}$, ϕ_u is admissible. (c.) If $u = o_{\mathcal{G}}$, ϕ_u semi-admissible.

5.2. Spatial, Frequency and Learning Indices

We introduce the key concept which defines the *spatial distance* among nodes. It is the length of the minimum spanning tree (MST) of the nodes.

Definition 1 (Spatial Index of Nodes) *Let $\mathfrak{n} \subseteq \mathcal{N}^{(d)}$. The spatial index of $\mathfrak{n}$ is defined to be*

$$\mathcal{S}(\mathfrak{n}) \equiv \min_{\mathfrak{n} \subseteq \mathcal{T} \leq \mathcal{G}^{(d)}} \sum_{uv \in \mathcal{E}(\mathcal{T})} \pi_{uv} = \min_{\mathfrak{n} \subseteq \mathcal{T} \leq \mathcal{G}^{(d)}} \sum_{uv \in \mathcal{E}(\mathcal{T})} \alpha_u \tag{19}$$

where $\mathfrak{n} \subseteq \mathcal{T} \leq \mathcal{G}^{(d)}$ means $\mathcal{T}$ is a sub-graph containing $\mathfrak{n}$. By default, $\mathcal{S}(\mathfrak{n}) = 0$ if $\mathfrak{n}$ contains only one or zero node.

Next, we define the spatial/frequency/learning indices for a multi-index $\boldsymbol{r} \in \boldsymbol{I} = I^{|\mathcal{N}_0^{(d)}|}$. Let $\boldsymbol{t}^{\boldsymbol{r}} : \boldsymbol{I} \to I$ be a monomial, where $\boldsymbol{r} : \mathcal{N}_0^{(d)} \to \mathbb{N}^{|\mathcal{N}_0^{(d)}|}$. We use $\mathfrak{n}(\boldsymbol{r}) = \{v \in \mathcal{N}_0^{(d)} : r_v \neq 0\}$ to denote the support of $\boldsymbol{r}$ and $\mathfrak{n}(\boldsymbol{r}; o_{\mathcal{G}}) = \mathfrak{n}(\boldsymbol{r}) \cup \{o_{\mathcal{G}}\}$.

Definition 2 (Spatial, Frequency and Learning Indices of $\boldsymbol{r}$) *We say $\boldsymbol{r} \in \mathbb{N}^{|\mathcal{N}_0^{(d)}|}$ is $\mathcal{G}^{(d)}$-learnable, or learnable for short, if there is a common ancestor node u of $\mathfrak{n}(\boldsymbol{r})$ such that ϕ_u is semi-admissible. We use $\mathfrak{A}(\mathcal{G}^{(d)}) \equiv \{\boldsymbol{r} \in \mathbb{N}^{\mathcal{N}_0^{(d)}} : \boldsymbol{r} \text{ is learnable}\}$. For $\boldsymbol{r} \in \mathfrak{A}(\mathcal{G}^{(d)})$, the spatial index, frequency index and the learning index are defined to be,*

$$\mathcal{S}(\boldsymbol{r}) \equiv \mathcal{S}(\mathfrak{n}(\boldsymbol{r}; o_{\mathcal{G}})), \quad \mathcal{F}(\boldsymbol{r}) \equiv \sum\nolimits_{v \in \mathcal{N}_0^{(d)}} \boldsymbol{r}_v \alpha_v \quad \textit{and} \quad \mathcal{L}(\boldsymbol{r}) \equiv \mathcal{S}(\boldsymbol{r}) + \mathcal{F}(\boldsymbol{r}), \tag{20}$$

resp. If $\boldsymbol{r} \notin \mathfrak{A}(\mathcal{G}^{(d)})$, we set $\mathcal{S}(\boldsymbol{r}) = \mathcal{F}(\boldsymbol{r}) = \mathcal{L}(\boldsymbol{r}) = +\infty$. Let $\mathcal{L}(\mathcal{G}^{(d)})$ denote the sequence of learning indices in non-descending order, i.e.

$$\mathcal{L}(\mathcal{G}^{(d)}) \equiv \Big(\mathcal{L}(\boldsymbol{r}) : \boldsymbol{r} \in \mathfrak{A}(\mathcal{G}^{(d)})\Big) \equiv (\cdots \leq r_j \leq r_{j+1} \leq \dots) \tag{21}$$

Finally, we specify the eigenfunctions of the kernels and the indices associated them. For each $u \in \mathcal{N}_0^{(d)}$, let $\{\overline{Y}_{r,l}\}_{l\in[N(d_u,r)],r\in\mathbb{N}}$ be the family of normalized spherical harmonics in $\overline{\mathbb{S}}_{d_u-1}$, where $N(d_u, r)$ is the number of degree r spherical harmonics in $\mathbb{S}_{d_u-1}$; see Sec. E.2 for more details about spherical harmonics. Define

$$\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) = \prod_{u\in\mathcal{N}_0^{(d)}} \overline{Y}_{\boldsymbol{r}_u,\boldsymbol{l}_u}(\boldsymbol{\xi}_u), \quad \boldsymbol{l} = (\boldsymbol{l}_u)_{u\in\mathcal{N}_0^{(d)}} \in [\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})] \equiv \prod_{u\in\mathcal{N}_0^{(d)}} [N(d_u, \boldsymbol{r}_u)] \tag{22}$$

for $\boldsymbol{\xi} = (\boldsymbol{\xi}_u) \in \boldsymbol{\mathcal{X}}$, where $\boldsymbol{d} = (d_u)_{u\in\mathcal{N}_0^{(d)}}$ and, for consistency, $\boldsymbol{d}_u = d_u$. The spatial, frequency and learning indices of $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ are $\mathcal{S}(\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}) \equiv \mathcal{S}(\boldsymbol{r})$, $\mathcal{F}(\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}) \equiv \mathcal{F}(\boldsymbol{r})$ and $\mathcal{L}(\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}) \equiv \mathcal{L}(\boldsymbol{r})$, resp.

5.3. Eigenspace Restructuring

The following is our main result. It describes an analytical relation between the architecture of a network and the eigenstructure of its inducing kernels in high dimensions.

Theorem 3 (Eigenspace Restructuring) *Assume* **Assumption-$\mathcal{G}$** *and* **Assumption-ϕ**. *We have the following eigen-decomposition for $\mathcal{K} = \mathcal{H}_{\mathcal{G}^{(d)}}$ or $\Theta_{\mathcal{G}^{(d)}}$. For $\boldsymbol{\xi}, \boldsymbol{\eta} \in \boldsymbol{\mathcal{X}}$*

$$\mathcal{K}(\boldsymbol{\xi},\boldsymbol{\eta}) = \sum\nolimits_{\boldsymbol{r}\in\mathbb{N}^{|\mathcal{N}_0^{(d)}|}} \lambda_{\mathcal{K}}(\boldsymbol{r}) \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}), \quad \textit{and} \quad \lambda_{\mathcal{K}}(\boldsymbol{r}) \sim d^{-\mathcal{L}(\boldsymbol{r})} \tag{23}$$

if $\boldsymbol{r} \neq 0$ and $|\boldsymbol{r}| \lesssim 1$.

The theorem says that the eigenfunctions of $\mathcal{K}$ are $\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\}$, whose eigenvalues are of order $\{d^{-\mathcal{L}(\boldsymbol{r})}\}$, which is determined by the learning index. When the DAGs are more *regular*, e.g. the network architectures are the CNNs, MLPs in Sec. B, then one can also prove that the dimension of eigenspaces with eigenvalues $\sim d^{-\mathcal{L}(\boldsymbol{r})}$ is $\sim d^{\mathcal{L}(\boldsymbol{r})}$ (see Lemma 10.)

Coupling with the observation in Eq. (4), Theorem 3 implies that within $t \sim d^r$ amount of time for gradient flow, when d is sufficiently large, only the eigenfunctions $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ with $\mathcal{L}(\boldsymbol{r}) \leq r$ can be learned. More precisely, let $\boldsymbol{\sigma}$ be the uniform (product) probability measure on $\boldsymbol{\mathcal{X}}$ and denote $L^p(\boldsymbol{\mathcal{X}}) \equiv L^p(\boldsymbol{\mathcal{X}}, \boldsymbol{\sigma})$. For $r \notin \mathcal{L}(\mathcal{G}^{(d)})$, define the projection operator to be

$$\boldsymbol{P}_{<r}(f) = \sum_{\boldsymbol{r}:\mathcal{L}(\boldsymbol{r})<r} \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})} \langle f, \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\rangle_{L^2(\boldsymbol{\mathcal{X}})} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\,,$$

and similarly for $\boldsymbol{P}_{>r}$. Let $\boldsymbol{\mathcal{F}} : L^2(\boldsymbol{\mathcal{X}}) \to L^2(\boldsymbol{\mathcal{X}})$ be the solution operator associated to the kernel gradient descent $\dot{h} = -\mathcal{K}(h - f)$ with $h_{t=0} = \mathbf{0}$ i.e. $h_t \equiv \boldsymbol{\mathcal{F}}_t(f) \equiv (\mathbf{Id} - e^{-\mathcal{K}t})f$, where

$$(\mathbf{Id} - e^{-\mathcal{K}t})f(\boldsymbol{x}) \equiv \int_{\bar{\boldsymbol{x}}\in\mathcal{X}} (1 - e^{-\mathcal{K}(\boldsymbol{x},\bar{\boldsymbol{x}})t}) f(\bar{\boldsymbol{x}}) d\bar{\boldsymbol{x}} \tag{24}$$

The following theorem says that when $t \sim d^r$, $\boldsymbol{\mathcal{F}}_t \approx \boldsymbol{P}_{<r}$.

Theorem 4 *Assume* **Assumption-$\mathcal{G}$** *and* **Assumption-ϕ** *and $\mathcal{K} = \mathcal{H}_{\mathcal{G}^{(d)}}$ or $\Theta_{\mathcal{G}^{(d)}}$. Let $r \notin \mathcal{L}(\mathcal{G}^{(d)})$ and $t \sim d^r$. Then for $0 < \epsilon < \inf\{|r - \bar{r}| : \bar{r} \in \mathcal{L}(\mathcal{G}^{(d)})\}$ and $f \in L^2(\boldsymbol{\mathcal{X}})$ with $\mathbb{E}_{\boldsymbol{\sigma}} f = 0$, we have*

$$\|\boldsymbol{\mathcal{F}}_t(\boldsymbol{P}_{<r}f) - \boldsymbol{P}_{<r}f\|_2^2 \lesssim e^{-d^\epsilon}\|\boldsymbol{P}_{<r}f\|_2^2 \quad \textit{and} \quad \|\boldsymbol{\mathcal{F}}_t(\boldsymbol{P}_{>r}f) - \boldsymbol{P}_{>r}f\|_2^2 \gtrsim e^{-d^{-\epsilon}}\|\boldsymbol{P}_{>r}f\|_2^2\,. \tag{25}$$

In words, in the infinite-training-data regime, within $t \sim d^r$ amount of time only the eigenfunctions $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ with $\mathscr{L}(\boldsymbol{r}) < r$ are learnable. The proof follows directly from the arguments in Sec. 3: an eigenfucntion $\boldsymbol{Y}_{\boldsymbol{r},\boldsymbol{l}}$ is learnable if $e^{-t\lambda_{\mathcal{K}}(\boldsymbol{r})} \sim e^{-d^{r-\mathscr{L}(\boldsymbol{r})}}$ is sufficiently small. Therefore, the learning index is the correct complexity measure that describes the compute efficiency mentioned in Sec.2.

5.4. Generalization: CNNs without pooling

Our next theorem concerns the finite-training-data regime, in which we will leverage a deep analytical result from Mei et al. (2021a) (Sec. 3 Theorem 4). We restrict the architectures to CNNs without pooling Eq. (15) which contain MLPs as special cases. For $X \subseteq \boldsymbol{\mathcal{X}}$, define the regressor,

$$\boldsymbol{R}_X(f)(x) = \mathcal{K}(x, X)\mathcal{K}(X, X)^{-1} f(X)\,.$$

Theorem 5 *Let $\boldsymbol{\mathcal{G}} = \{\mathcal{G}^{(d)}\}_d$, where each $\mathcal{G}^{(d)}$ is a DAG associated to a CNN Eq. (15). Let $r \notin \mathscr{L}(\mathcal{G}^{(d)})$ be fixed. Assume* **Assumption-ϕ** *and* **Assumption-$\boldsymbol{\mathcal{G}}$**. *Let $f \in L^2(\boldsymbol{\mathcal{X}})$ with $\mathbb{E}_{\boldsymbol{\sigma}} f = 0$. Then for $\epsilon > 0$,*

$$\left| \|\boldsymbol{R}_X(f) - f\|^2_{L^2(\boldsymbol{\mathcal{X}})} - \|\boldsymbol{P}_{>r}(f)\|^2_{L^2(\boldsymbol{\mathcal{X}})} \right| = c_{d,\epsilon} \|f\|^2_{L^{2+\epsilon}(\boldsymbol{\mathcal{X}})}, \tag{26}$$

where $c_{d,\epsilon} \to 0$ in probability as $d \to \infty$ over $X \sim \boldsymbol{\sigma}^{[d^r]}$.

In words, with $[d^r]$ many training examples where $r \notin \mathscr{L}(\mathcal{G}^{(d)})$, the NNGP kernel and the NTK are able to learn all $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ with $\mathscr{L}(\boldsymbol{r}) < r$ but not any eigenfunctions with $\mathscr{L}(\boldsymbol{r}) > r$. Therefore, the learning index is the correct complexity measure that describes the data efficiency mentioned in Sec.2.

5.5. Generalization and training efficiency of CNNs with pooling

In the appendix Sec. H, we prove similar results for CNNs with pooling. Comparing to no pooling, global average pooling (Theorem 13) improves the data-efficiency by a factor of w, the window size of pooling. However, the training efficiency in the infinite-data-regime is the same (Theorem 14) for CNNs with and without pooling. We provide empirical support of these claims in Sec. C.3 by conducting experiments on synthetic data using toy models and on natural data (ImageNet) using ResNet50.

Finally, the requirement $r \notin \mathscr{L}(\mathcal{G}^{(d)})$ (aka the non-critical scaling regime), makes the analysis simpler. When $r \in \mathscr{L}(\mathcal{G}^{(d)})$ (aka the critical scaling regime), the generalization error, as a function of training set size $|X|$, can be non-monotonic and the whole sample-wise generalization curves can exhibit multiple-descent behaviors; see follow-up work Xiao and Pennington (2022) and Misiakiewicz (2022); Hu and Lu (2022).

6. Interpretation of the Main Results

We use Theorem 5 and Theorem 4 to explain why better architectures lead to better performance. In particular, we show how *locality*, *hierarchy*, *model scaling*[1] work together to progressively improve learnability and generalization of the networks. In Sec. C, we verify our theoretical claims

1. By model scaling, we mean increasing the depth and decreasing the filter sizes simultaneously. This is similar to increasing both the depth and the resolutions of the input images.

empirically for both kernels and finite-width networks (trained by SGD + Momentum) for datasets and networks of practical sizes.

We say r is the budget index if (1) (**finite-data-regime**) the training set size $m \sim d^r$, or (2) (**infinite-data-regime**) the training set $X = \boldsymbol{\mathcal{X}}$ and the total number of training steps/time $t \sim d^r$. Assume $\boldsymbol{\mathcal{X}} = (\overline{\mathbb{S}}_{p-1})^n$, where $p = d^{\alpha_p}$, $n = d^{\alpha_n}$, $\alpha_p + \alpha_n = 1$ and d is sufficiently large. Recall that $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ is learnable if $\mathbb{S}(\boldsymbol{r}) + \mathfrak{F}(\boldsymbol{r}) < r$. Increasing the resolutions of $\{\mathbb{S}(\boldsymbol{r})\}_{\boldsymbol{r}\in\mathbb{N}^n}$ and $\{\mathfrak{F}(\boldsymbol{r})\}_{\boldsymbol{r}\in\mathbb{N}^n}$, i.e. finely partitioning the eigenspaces, is crucial for improving learnability and generalization. For simiplicity, we consider order $h \geq 2$ interaction between two *mini-patches* $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in \overline{\mathbb{S}}_{p-1}$, where $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ and $i \neq j \in [n]$. Namely, we assume $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{x}) = \phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for some degree h homogeneous polynomial ϕ. We slightly abuse the notation to denote $\phi = \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$. For concreteness, we choose $r = 2.8$, $\alpha_p = \alpha_n = 0.5$.

Baseline: MLPs. Regardless of the number of layers and the spatial locations of i and j, we have $\mathfrak{L}(\phi) = \mathfrak{F}(\phi) = h$. I.e. we need at least d^h (samples/time) to learn ϕ and only those ϕ with $h < r$ are learnable, i.e. $h = 2$. Therefore, MLPs can model low-frequency interactions but they are highly inefficient in modeling any type of high-frequency interactions; see the top panel in Fig. 3. Moreover, making the network deeper won't help (Fig. 7).

+Locality: 1-layer CNNs. Assume the first layer of the network is a convolutional layer with kernel size and strides p, the remaining layers are dense layers. Then the frequency index of ϕ is $h\alpha_p$ and the spatial index is $2\alpha_n$ (because there are two mini-patches i/j) and the learning index is $\mathfrak{L}(\phi) = 2\alpha_n + h\alpha_p = h - (h-2)(1-\alpha_p)$. There is a gain of $(h-2)(1-\alpha_p)$ over MLPs and we can model $h = 3$ degree of interactions (since $2\alpha_p + 3\alpha_n = 2.5 < 2.8 = r$.) Note that localities improve the resolutions of the frequency indices from $\{k : k \in \mathbb{N}\}$ (without localities) to $\{\alpha_p k : k \in \mathbb{N}\}$ (with localities.)

+Locality + Hierarchy: deep CNNs. We replace the dense layers in the above 1-layer CNN with L convolutional layers with filter sizes and strides $k = n^{1/L}$ (plus dense layers afterward.) The frequency index of ϕ is the same $h\alpha_p$. However, spatial index can vary from $(L+1)\alpha_k$ to $2L\alpha_k$ (where $\alpha_k = \alpha_n/L$) arithmetically, depending on the spatial distance between $i, j \in [k]^L \cong [n]$. Therefore the learning index is $\mathfrak{L}(\phi) = h\alpha_p + l\alpha_n/L$, where $l \in [L+1, 2L] \cap \mathbb{N}$ is the number of edges in the DAG connecting i and j to the output nodes; See Sec.D. The gain over 1-layer CNNs is $(2L-l)\alpha_n/L$. Choosing $L = 2$, we can model $h = 4$ interactions when $l = L+1 = 3$ since $4\alpha_p + 3\alpha_n/2 = 2.75 < r$, but not any $h \geq 5$ or $l = 2L$ (longer-range) with $h \geq 4$ interactions.

+Locality + Hierarchy + Model Scaling: HR-CNNs. It is clear now, to capture as many types of interactions as possible, we should choose α_p (i.e. patch size p) and α_k (i.e. filter sizes k) as small as possible and simultaneously, L as large as possible. E.g., choosing $\alpha_p = 0.1 = \alpha_k = 0.1$ (i.e. $\alpha_n = 0.9$ and $L = 9$), the highest order (very short range) interactions that can be modeled are $h = 17$ (solve $h\alpha_p + (L+1)\alpha_n/L < 2.8$.) In contrast, to model this interaction ($h = 17$) the above architectures require sample complexity to be at least d^{17^+} (MLPs), $d^{9.5^+}$ (1-layer CNNs with $\alpha_p = \alpha_n = 0.5$), and $d^{9.25^+}$ (2-layer CNNs with $\alpha_p = \alpha_n = 0.5$.) For practical networks, e.g., ResNet (He et al., 2016), the filters/patches are already quite small, and there isn't much room to reduce them. Equivalently, one increases the resolution of the input images instead (also need to increase the compute), i.e., increasing d (Tan and Le, 2019).

In Sec. C, we provide empirical verification of the above claims by conducting experiments using both kernel regression (NNGP/NTK) and SGD + Momentum on finite-width networks in the

strong finite-size correction regime, in which the input dimension is $d = 256$ and patch size is only $p = 4$ (i.e., $d = p^4$). In both sets of experiments, we see good agreement between theoretical predictions and finite-size simulations. Fig. 3 demonstrates the learning separation between MLPs (baseline, top panel), deep CNNs (+Locality + Hierarchy, middle panel) and HR-CNNs (+Locality + Hierarchy + Model Scaling, bottom channels.) in terms of both data-efficiency and training-efficiency.

7. Additional Related work

Bietti (2021) studies the approximation and learning properties of 2-3 layer CNN kernels via the lens of RKHS and demonstrates that a 3-layer CNN kernel can reach 88.3% validation accuracy on CIFAR-10, matching the performance of a 10-layer Myrtle kernel Shankar et al. (2020). The perspective of Bietti (2021) is different from our: Bietti (2021) aims for precise characterizations of the induced regularities (aka RKHS norms) by pooling, and interactions between patches, while we focus on the relation between architectures and the induced eigenstructures. Malach and Shalev-Shwartz (2020) and Li et al. (2020) study the algorithmic benefits of shallow CNNs and show that they outperform MLPs in certain tasks. Xiao and Pennington (2021) and Favero et al. (2021) study the benefits of locality in S-CNNs and argue that localities are the key ingredient to defeat the curse of dimensionality. However, the models they studied are essentially a linear combination of one-layer MLPs, each defined on a batch. The models have very limited expressivity and can not capture any nonlinear interactions between different patches. Mei et al. (2021b) and several papers mentioned above also study the benefits of pooling in (S-)CNNs in terms of data efficiency. Mei et al. (2021b) is the first to rigorously prove the benefits and limitations of pooling when the patch size is equal to the spatial dimensions. Their conclusion is similar to that of Theorem 13: pooling improves data efficiency by a factor of the pooling size. In addition, we show that (Theorem 14) pooling does not improve training efficiency for D-CNNs, extending a result from Xiao and Pennington (2021) which concerns S-CNNs. Finally, Scetbon and Harchaoui (2020) also studies the eigenstructures of certain CNN kernels without pooling. Their kernels can be considered as a particular case of the NNGP kernels, where the associated networks have only one convolutional layer and multiple dense layers with polynomial activations. As such, the role of hierarchical localities is not studied there. Moreover, the asymptotic limit of interest is different: Scetbon and Harchaoui (2020) treats the dimension number d as a constant while our analysis focuses on the regime $d \to \infty$. In sum, the key contribution that sets the current work apart from existing work is that our work offers a precise mathematical characterization of the fundamental role of architectures in (infinite-width) networks through a space-frequency analysis. In particular, we rigorously prove that D-CNNs (more precisely, hierarchical localities) are able to break the curse of dimensionality, and scaling improves performance in both finite and infinite data regimes.

8. Discussion and Conlcusion

We establish a precise relation between the architectures of networks, the eigenstructures of the inducing kernels, and generalization of the corresponding kernel machines in the high-dimensional setting. We show that deep convolutional networks restructure the eigenspaces of the inducing kernels, which empowers them to learn a dramatically broader class of functions, covering a wide range of space-frequency combinations without extra data. We prove that infinite-width convolutional net-

works is able to overcome the curse of dimensionality. On the practical side, our results suggest that using images with higher resolutions, networks with more layers and smaller filter sizes, and increasing the training set size[2] can improve the performance of the models, which have already been observed by practitioners (Tan and Le, 2019; Kaplan et al., 2020).

We believe our framework can be extended to study architectural inductive biases for other families of topologies, such as RNNs, GNNs, and even self-attention. It is expected that the exact mathematical formulation of the *learning index* will depend on the structure of the model of interest. We have not covered the learning dynamics of SGD in the feature learning regime, which is a very challenging topic. In addition, it is of great interest and importance to study the combined effect of SGD and the architectural inductive biases in the future. Our results suggest that better architectures (such as D-CNNs, HR-CNNs) provide a better search space for SGD, in the sense that the initial descent direction, namely, the NTK, biases the model towards learning functions with lower complexity (defined by the learning index). Another limitation of the current paper lies in the strong assumption on the input space, which requires the data to be drawn *uniformly* from a product of hyperspheres. There are two orthogonal directions to relax this assumption. One is to restrict the support of each patch in some low dimensional compact manifold, and we expect tools from studying eigenfunctions on Riemannian manifolds can be helpful here (Zelditch, 2017). The other one is to impose dependence between patches, e.g., *spatially close* patches have strong correlations.

Finally, our results also suggest the need for rethinking approximation properties of *finite-width* neural networks through the lens of *space-frequency* analysis, at least for convolution-based architectures. While the frequency structures, namely, smoothness of the target functions in terms of the Sobolev norm, have been broadly studied during the last decades (Cucker and Smale, 2001; von Luxburg and Bousquet, 2004; Bach, 2017), spatial structures are mostly overlooked. Consequently, the required widths of the networks to approximate certain functions often grow rapidly (often polynomially) in the input dimensions (Eldan and Shamir, 2016; Daniely, 2017), i.e., the growth rates of the widths and the total number of parameters are cursed by the dimensionality. Nevertheless, the widths used in SotA models are much smaller than the input dimensions, e.g., the largest width in ResNets (He et al., 2016) is $2048 \approx d^{0.64}$, where $d = 224^2 \times 3$ is the (default) input dimension of ImageNet used by the models. We conjecture that hierarchical localities are also the key ingredient to break the curse of dimensionality for approximation in finite-width neural networks, bringing down the growth rate of the widths to a more realistic range.

Acknowledgments and Disclosure of Funding

We thank Atish Agarwala, Ben Adlam, Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak and Sam Schoenholz for discussions and feedbacks on the project. We are also grateful to Yasaman Bahri and the anonymous reviewers for feedback and suggestions on an earlier draft of this work. We acknowledge the Python community (Van Rossum and Drake Jr, 1995) for developing the core set of tools that enabled this work, including NumPy (van der Walt et al., 2011), SciPy (Virtanen et al., 2020), Matplotlib (Hunter, 2007), Pandas (Wes McKinney, 2010), Jupyter (Kluyver et al., 2016), JAX (Bradbury et al., 2018b), Neural Tangents (Novak et al., 2020), FLAX (Heek et al., 2020), Tensorflow datasets (TFD) and Google Colaboratory (Research). This work was performed at and funded by Google. No third party funding was used.

2. This may seem obvious, but our results suggest that networks with modeling scaling could *smoothly* utilize the increment of training data since the set of learning indices has higher resolutions.

References

TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

William Beckner. Inequalities in fourier analysis. *Annals of Mathematics*, 102(1):159–182, 1975.

William Beckner. Sobolev inequalities, the poisson semigroup, and analysis on the sphere sn. *Proceedings of the National Academy of Sciences*, 89(11):4816–4819, 1992.

Alberto Bietti. Approximation and learning with deep convolutional models: a kernel perspective, 2021.

Alberto Bietti and Francis Bach. Deep equals shallow for relu networks in kernel regimes. *arXiv preprint arXiv:2009.14397*, 2020.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2006. ISBN 978-0-387-31073-2.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018a. URL http://github.com/google/jax.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018b. URL http://github.com/google/jax.

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

Felipe Cucker and Stephen Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2001.

Amit Daniely. Depth separation for neural networks. In *Conference on Learning Theory*, pages 690–696. PMLR, 2017.

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.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

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.

Arturo Deza, Qianli Liao, Andrzej Banburski, and Tomaso Poggio. Hierarchically compositional tasks and deep convolutional networks. *arXiv preprint arXiv:2006.13915*, 2020.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks, 2016.

Alessandro Favero, Francesco Cagnetta, and Matthieu Wyart. Locality defeats the curse of dimensionality in convolutional teacher-student scenarios, 2021.

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

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 9108–9118, 2019.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax.

Hong Hu and Yue M Lu. Sharp asymptotics of kernel ridge regression beyond the linear regime. *arXiv preprint arXiv:2205.06798*, 2022.

J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007.

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

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.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

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.

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

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.

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.

Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets? *CoRR*, abs/2010.08515, 2020. URL https://arxiv.org/abs/2010.08515.

Eran Malach and Shai Shalev-Shwartz. Computational separation between convolutional and fully-connected networks. *CoRR*, abs/2010.01369, 2020. URL https://arxiv.org/abs/2010.01369.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018a.

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*, 2018b.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Generalization error of random features and kernel methods: hypercontractivity and kernel matrix concentration. *arXiv preprint arXiv:2101.10588*, 2021a.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Learning with invariances in random features and kernel models. *arXiv preprint arXiv:2102.13219*, 2021b.

Theodor Misiakiewicz. Spectrum of inner-product kernel matrices in the polynomial regime and multiple descent phenomenon in kernel ridge regression. *arXiv preprint arXiv:2204.10425*, 2022.

Ashley Montanaro. Some applications of hypercontractive inequalities in quantum information theory. *Journal of Mathematical Physics*, 53(12):122206, 2012.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, 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*, 2019a. URL https://openreview.net/forum?id=B1g30j0qF7.

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*, 2019b.

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*, 2019c.

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

Yaniv Ovadia, Emily Fertig, J. Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

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.

Google Research. Google colab. URL https://colab.research.google.com/.

Meyer Scetbon and Zaid Harchaoui. Harmonic decompositions of convolutional networks. In *International Conference on Machine Learning*, pages 8522–8532. PMLR, 2020.

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

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.

Vaishaal Shankar, Alex Chengyu Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, 2020.

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.

Jascha Sohl-Dickstein, Roman Novak, Samuel S. Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization, 2020.

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

S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, 2011.

Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

M. Alex O. Vasilescu, Eric Kim, and Xiao S. Zeng. Causalx: Causal explanations and block multilinear factor analysis, 2021.

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.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *J. Mach. Learn. Res.*, 5:669–695, 2004.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

Lechao Xiao and Jeffrey Pennington. What breaks the curse of dimensionality in deep learning?, 2021. URL https://openreview.net/forum?id=KAV7BDCcN6.

Lechao Xiao and Jeffrey Pennington. Precise learning curves and higher-order scaling limits for dot product kernel regression. *arXiv preprint arXiv:2205.14846*, 2022.

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.

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.

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

Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks, 2020.

Steve Zelditch. *Eigenfunctions of the Laplacian on a Riemannian manifold*, volume 125. American Mathematical Soc., 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Appendix A. Organization of the Appendix.

The appendix is organized as follows.

- Sec. B: three concrete examples to unpack the notations in Sec. 4.
- Sec. C: the main experimental results of the paper. Additional experiments can be found in Sec. J.
- Sec. D: details about the DAGs and the eigenfunctions used in the experimental section, and detailed examples about how to compute the spatial, frequency and learning indices.
- Sec. E: the proof of the eigenspace restructuring theorem.
- Sec. F and Sec. H: the proof of the generalization bound for CNNs without and with pooling, resp. We briefly recap the main analytic tools used in the proof in Sec. I.
- Sec. K: the code for the architectures used in the synthetic dataset experiments.

Appendix B. Three Examples: MLPs, S-CNNs and D-CNNs.

To unpack the notation in Sec. 4, we consider three concrete examples: an L-hidden layer MLP, a shallow convolutional network (S-CNN) that contains only one convolutional layer and a deep convolutional network (D-CNN) that contains $(1 + L)$ convolutional layers. The architectures are

$$\textbf{MLP:} \quad [\textit{Input}] \to [\textit{Dense-Act}]^{\otimes L} \to [\textit{Dense}] \tag{27}$$

$$\textbf{S-CNN:} \quad [\textit{Input}] \to [\textit{Conv}(p)\textit{-Act}] \to [\textit{Flatten-Dense}] \tag{28}$$

$$\textbf{D-CNN:} \quad [\textit{Input}] \to [\textit{Conv}(p)\textit{-Act}] \to [\textit{Conv}(k)\textit{-Act}]^{\otimes L} \to [\textit{Flatten-Dense-Act}] \to [\textit{Dense}] \tag{29}$$

where p/k is the filter size of the first/hidden layers and $Dense/Conv/Act/Flatten$ means an dense / convolutional / activation / flattening layer. We choose the stride to be the same as the size of the filter for all convolutional layers and choose *flattening* as the readout strategy rather than pooling. See Fig. 6 for the DAGs associated to a 4-layer MLP, a (1 + 1)-layer CNN (with $p = k = d^{\frac{1}{2}}$) and a (1+3)-layer CNN (with $p = k = d^{\frac{1}{4}}$).

MLPs. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a linked list with $(L + 2)$ nodes, including the input/output nodes. Let $\mathcal{L}_{uv} \in \mathbb{R}^{n_u \times n_v}$, where $n_{u/v} = \dim(\mathbb{H}_{u/v})$ and the activations of the input/output nodes be the identity function. Then $\mathcal{N}_{\mathcal{G}}$ represents a L-hidden-layer MLP. In addition, let u and v be the nodes in the l- and $(l-1)$-th layers and let $\mathcal{L}_{uv}$ be initialized iid as

$$\mathcal{L}_{uv} = \frac{1}{\sqrt{n_v}} \left(\omega_{uv,ii'}\right)_{i\in[n_u],i'\in[n_v]} \equiv \frac{1}{\sqrt{n_v}} \left(\omega_{ii'}^{(l)}\right)_{i\in[n_u],i'\in[n_v]}, \quad \omega_{uv,ii'} \equiv \omega_{ii'}^{(l)} \sim \mathcal{N}(0,1)\,. \tag{30}$$

Then the MLP can be written recursively as

$$\mathcal{N}_u(\boldsymbol{x}) = \phi_u(\mathcal{L}_{uv}(\mathcal{N}_v(\boldsymbol{x}))) = \phi_u\left(\frac{1}{\sqrt{n_v}}\omega^{(l)}\mathcal{N}_v(\boldsymbol{x})\right) \tag{31}$$

Let $\theta = (\omega_{uv,ii'} : i \in [n_u], i' \in [n_v], uv \in \mathcal{E})$ denote the collection of all trainable parameters. Let $\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'} = \boldsymbol{x}^T\boldsymbol{x}'/n_u$ for $u \in \mathcal{N}_0$ and $n_v \to \infty$ for all hidden nodes, then the outputs of $\mathcal{N}_\mathcal{G}(\boldsymbol{\mathcal{X}})$ converge weakly to the GP $\mathcal{GP}(0, \mathcal{K}_\mathcal{G}(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'})_{\boldsymbol{x},\boldsymbol{x}'\in\boldsymbol{\mathcal{X}}})$ and $\Theta_\mathcal{G}(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'})_{\boldsymbol{x},\boldsymbol{x}'\in\boldsymbol{\mathcal{X}}}$ is the NTK in the sense

$$\mathbb{E}\mathcal{N}_\mathcal{G}(\boldsymbol{x})\mathcal{N}_\mathcal{G}(\boldsymbol{x}') \xrightarrow{\text{in prob.}} \mathcal{K}_\mathcal{G}(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'}) \quad \text{and} \quad \langle \nabla_\theta \mathcal{N}_\mathcal{G}(\boldsymbol{x}), \nabla_\theta \mathcal{N}_\mathcal{G}(\boldsymbol{x}')\rangle \xrightarrow{\text{in prob.}} \Theta_\mathcal{G}(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'}). \tag{32}$$

Indeed, note that $\deg(u) = 1$ for all $u \notin \mathcal{N}_0$. Eq. (11) and Eq. (12) become

$$\mathcal{K}_u(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'}) = \phi_u^*(\mathcal{K}_v(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'})) \quad \text{and} \quad \Theta_u(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'}) = \dot{\phi}_u^*(\mathcal{K}_v(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'}))(\mathcal{K}_v(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'}) + \Theta_v(\boldsymbol{t}_{\boldsymbol{x},\boldsymbol{x}'})) \tag{33}$$

which are exactly the recursive formulas for the NNGP kernel and NTK for MLPs; see e.g. Sec.E in Lee et al. (2019).

S-CNN. The input $\boldsymbol{\mathcal{X}} = (\overline{\mathbb{S}}_{p-1})^w \subseteq \mathbb{R}^d$, where p is the patch size, w is the number of patches, $d = pw$ is the dimension of the inputs. Here, the inputs have been *pre-processed* by a patch extractor and then by a normalization operator. In words, the S-CNN has one convolutional layer with filter size p, followed by an activation function ϕ (e.g., Relu), and finally by a *flatten-dense* readout layer. Mathematically, by letting $n \in \mathbb{N}$ be the number of channels in the hidden layer, the output (i.e., logit) is given by

$$\textbf{Convolution + Activation:} \qquad z_{ij}(\boldsymbol{x}) = \phi\left(p^{-\frac{1}{2}} \sum_{i'\in[p]} \omega_{j,i'}^{(1)} \boldsymbol{x}_{i,i'}\right) \quad \text{for} \quad i \in [w], j \in [n] \tag{34}$$

$$\textbf{Flatten + Dense:} \qquad f(\boldsymbol{x}) = (wn)^{-\frac{1}{2}} \sum_{i\in[w],j\in[n]} \omega_{ij}^{(2)} z_{ij}(\boldsymbol{x})\,, \tag{35}$$

where $\omega_{j,i'}^{(1)}$ and $\omega_{ij}^{(2)}$ are the parameters of the first and readout layers, resp.

We can associate a DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ to the above S-CNN. Let the input, hidden and output nodes be $\mathcal{N}_0 = \{0\} \times [w], \mathcal{N}_1 = \{1\} \times [w]$ and $\mathcal{N}_2 = \{o_\mathcal{G}\} = \{(2)\}$, resp. and $\mathcal{N} = \mathcal{N}_0 \cup \mathcal{N}_1 \cup \mathcal{N}_2$. Moreover, $uv \in \mathcal{E}$ if $u = o_\mathcal{G}$ and $v \in \mathcal{N}_1$ or $u = (1, i) \in \mathcal{N}_1$ and $v = (0, i) \in \mathcal{N}_0$. Let $\mathbb{H}_v = \mathbb{R}^p$ for $v \in \mathcal{N}_0$, $\mathbb{H}_u = \mathbb{R}^n$ if $u \in \mathcal{N}_1$ and $\mathbb{H}_{o_\mathcal{G}} = \mathbb{R}$. The associated linear operators are given by

$$\mathcal{L}_{uv} = p^{-\frac{1}{2}} \left(\omega_{j,i'}^{(1)}\right)_{j\in[n],i'\in[p]} \in \mathbb{R}^{n\times p} \qquad \text{for} \quad (u, v) \in \mathcal{N}_1 \times \mathcal{N}_0, \quad uv \in \mathcal{E}. \tag{36}$$

$$\mathcal{L}_{o_\mathcal{G}v} = (wn)^{-\frac{1}{2}} \left(\omega_{ij}^{(2)}\right)_{j\in[n]} \in \mathbb{R}^n \qquad \text{if} \quad v = (1, i) \in \mathcal{N}_1 \tag{37}$$

Note that the weights are shared in the first layer but not in the readout layer (i.e., the network has no pooling layer). To compute the NNGP kernel and NTK, we initalize all parameters $\omega_{j,i'}^{(1)}$ and $\omega_{ij}^{(2)}$ with iid Gaussian $\mathcal{N}(0, 1)$. Letting $n \to \infty$ and denoting $\boldsymbol{t}_v = \boldsymbol{x}_v^T\boldsymbol{x}_v'/p$ and $\boldsymbol{t} = (\boldsymbol{t}_v)_{v\in\mathcal{N}_0}$, we have

$$\mathcal{K}_\mathcal{G}(\boldsymbol{t}) = \fint_{v\in\mathcal{N}_0} \phi^*(\boldsymbol{t}_v) \quad \text{and} \quad \Theta_\mathcal{G}(\boldsymbol{t}) = \fint_{v\in\mathcal{N}_0} \phi^*(\boldsymbol{t}_v) + \dot{\phi}^*(\boldsymbol{t}_v) \tag{38}$$

D-CNN. The input space is $\boldsymbol{\mathcal{X}} = (\overline{\mathbb{S}}_{p-1})^{w\times k^L} \subseteq \mathbb{R}^{w\times k^L\times p}$, where p is the patch size of the input convolutional layer, k is the filter size in *hidden* convolutional layers, L is the number of *hidden* convolutional layers and w is the spatial dimension of the layer before flattening. The total

dimension of the input is $d = p \cdot k^L \cdot w$, and the number of input nodes is $|\mathcal{N}_0| = k^L \cdot w$. Since the stride is equal to the filter size for all convolutional layers, the *spatial* dimension is reduced by a factor of p in the first layer, a factor of k by each hidden convolutional layer, and is reduced to 1 by the *Flatten-Dense-Act* layer. Similar to S-CNNs, one can associate a DAG to a D-CNN. Briefly, the input layer has $k^L \times w$ nodes and is reduced by a factor of k by each convolutional layer. The layer before *flattening*, the second last layer and the output layer have w, 1, and 1 nodes, resp. More precisely, we can identify the nodes using tuples.

- For $j = 0$, i.e., the input layer, $\mathcal{N}_0 = \{0\} \times [w] \times [k]^L$.
- For $1 \le j \le L+1$, $\mathcal{N}_j = \{j\} \times [w] \times [k]^{L-j+1}$.
- The second last and output layers are $\mathcal{N}_{L+2} = \{(L+2)\}$ and $\mathcal{N}_{L+3} = \{(L+3)\} \equiv \{o_{\mathcal{G}}\}$.

Note that the first index of a tuple specifies the layer index of the node, i.e. if $u = (j, \dots)$ then $u \in \mathcal{N}_j$. The remaining indices specify its spatial location in that layer. To define the edges $\mathcal{E}$, we only need to specify the parents of the nodes.

- For $j = 0$, the parent of $v = (0, k_0, k_1, \dots, k_L) \in \mathcal{N}_0$ is $u = (1, k_0, k_1, \dots, k_L) \in \mathcal{N}_1$.
- For $1 \le j \le L$, the parent of $v = (j, k_0, k_1, \dots, k_{L-j+1}) \in \mathcal{N}_j$ is $u = (j{+}1, k_0, k_1, \dots, k_{L-j}) \in \mathcal{N}_{j+1}$.
- For $j = L+1$, the parent of $v = (L+1, k_0) \in \mathcal{N}_{L+1}$ is $u = (L+2)$, whose parent is $o_{\mathcal{G}} = (L+3)$.

We write $\mathcal{E} = \cup_{1 \le j \le L+3} \mathcal{E}_j$, where $\mathcal{E}_j$ is the collection of edges between the $(j-1)$-th and j-th layer. Next, we define the neural network associated to this D-CNN. For $u \in \mathcal{N}_j$, $\mathbb{H}_u = \mathbb{R}^{n_j}$, where $n_0 = p$ is the size of the patch, n_j is the number of channels in the j-th hidden layer for $1 \le j \le L+1$, and n_{L+2} is the number of features in the second last layer and $n_{L+3} = 1$. Thus $n_u = n_j$ if $u \in \mathcal{N}_j$. For $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$, if $u = (0, k_0, k_1, \dots, k_L) \in \mathcal{N}_0$, let

$$\mathcal{N}_u(\boldsymbol{x}) \equiv \boldsymbol{x}_u \equiv \boldsymbol{x}_{k_0, k_1, \dots, k_L} \in \overline{\mathbb{S}}_{p-1} \tag{39}$$

and otherwise, $u \in \mathcal{E}_j$ for some $j \ge 1$ and let

$$\mathcal{N}_u(\boldsymbol{x})_i = \phi_u \left((\deg(u) n_v)^{-\frac{1}{2}} \sum_{v : uv \in \mathcal{E}_j} \sum_{i' \in [n_v]} \omega_{uv, ii'} \mathcal{N}_v(\boldsymbol{x})_{i'} \right) \quad \text{for} \quad i \in [n_u]. \tag{40}$$

Here, for $u \in \mathcal{N}_j$, the degree $\deg(u) = p, k, w$ and 1 if $j = 1$, $2 \le j \le L+1$, $j = L+2$ and $j = L+3$ resp., and the parameters are usually initialized with iid $\omega_{uv, ii'} \sim \mathcal{N}(0, 1)$ and the shape of ω_{uv} is $(\deg(u), n_v, n_u)$. It is more appropriate to call the current network a locally-connected network, or a "convolutional network" *without* weight-sharing, as the parameters ω_{uv} depend on the node u and are not shared within the same layer. To make it a true convolutional network, we enforce weight-sharing by setting $\omega_{uv, ii'} = \omega_{u'v, ii'} \equiv \omega^{(j)}_{v, ii'}$ if $u, u' \in \mathcal{N}_j$ for some j.

Denoting $\boldsymbol{t}_u = \boldsymbol{x}_u^T \boldsymbol{x}'_u / p$ for $u \in \mathcal{N}_0$ and $\boldsymbol{t} = (\boldsymbol{t}_u)_{u \in \mathcal{N}_0}$. As $\min\{n_j\}_{1 \le j \le L+2} \to \infty$, we have Eq. (32) and the NNGP kernel and the NTK associated to this network are given by Eq. (11) and Eq. (12). For formal proofs, see Daniely et al. (2016); Novak et al. (2019c); Yang (2019). This is true

for both convolutional networks and locally-connected networks, as long as the architecture contains no pooling (Novak et al., 2019c). For a convolutional network with a global average pooling layer (GAP), the kernel computations need to be modified to capture the translation invariance from GAP. See Sec. H for more details.

Appendix C. Experiments

There are many practical consequences due to the above theorems. For Theorem 3 and Theorem 5, we focus on two of them which are about *the impact of architectures to learning / generalization*:

1. **Order of Learning (Fig. 1 Green Arrow.)** The order of learning is restructured from frequency-based (MLPs) to space-and-frequency-based (CNNs).

2. **Learnability (Fig. 1 Cells under the budget line.)** With the same budget index, MLP-Learnable $\subsetneq$ D-CNN-Learnable $\subsetneq$ HR-CNN-Learnable. Moreover, the set differences between these learnable sets are captured as in Sec.6.

Here D-CNN stands for deep convolutional neural networks and HR-CNN stands for high-resolutions convolutional neural networks, which are deeper CNNs with small patch size and filter sizes.

For Theorem 13 and Theorem 14, we focus on the training and data efficiency of *CNN+GAP* (GAP for short) and *CNN+Flatten* (Flatten for short):

1. When the dataset size is sufficiently large, GAP and Flatten are equally efficient.

2. When the dataset size is relatively small compared to the learning index, GAP is more data-efficient.

Overall, we see excellent agreements between predictions from our theorems and experimental results from both practical-size networks and kernel methods using NNGP/NT kernels, even when $d = 256$ is moderate-size. We detail the setup, results, corrections, etc., for the experiments below.

C.1. Experimental Setup

Set $d = p^4$ and the input $\boldsymbol{\mathcal{X}} = (\overline{\mathbb{S}}_{p-1})^{p^3} \subseteq \mathbb{R}^{p^4}$, where $p \in \mathbb{N}$. Note that $\alpha_p = 1/4$. The task is learning a function $Y \in L^2(\boldsymbol{\mathcal{X}})$ by minimizing the MSE, where

$$Y = \mathbf{Y_1} + \mathbf{Y_2} + \mathbf{Y_3} + \mathbf{Y_4} + \mathbf{Y_5} + \mathbf{Y_5^*} + \mathbf{Y_6} + \mathbf{Y_7} \tag{41}$$

and each eigenfunction Y_i is a normalized so that $\|Y_i\|_2^2 = \|Y\|_2^2/8 = 1$. See Sec. D for the expressions of these functions.

We optimize finite-width networks by *SGD+Momentum* and infinite-width networks (NNGP and NTK) by kernel regression. We investigate three types of architectures: (1) MLP$^{\otimes 4}$, a four hidden layer MLP; (2) Conv$(p^2)^{\otimes 2}$, a "deep" CNN with filter size/stride $k = p^2$, and (3) Conv$(p)^{\otimes 4}$, a "HR"-CNN with filter size/stride $k = p$. See Fig.1 for a visualization of the associated DAGs. There is an activation ϕ in each *hidden* layer, which is chosen so that ϕ^* is the Gaussian kernel. For the CNNs, the readout layer(s) is *Flatten-Dense-Act-Dense*. We provide the code for these architectures in Sec. K. We carefully chose the eigenfunctions $\{Y_i\}$ so that they cover a wide range of space-frequency combinations $(\mathcal{S}(Y_i), \mathcal{F}(Y_i))$ w.r.t. Conv$(p)^{\otimes 4}$. Under Conv$(p)^{\otimes 4}$, the corresponding learning indices are $\mathcal{L}(Y_i) = \mathcal{S}(Y_i) + \mathcal{F}(Y_i) = 3\alpha_p + i\alpha_p = (3+i)/4$. For the learning indices

of Y_i under $\text{Conv}(p^2)^{\otimes 2}$ or $\text{MLP}^{\otimes 4}$, see the legends in Fig.3. The purpose of doing so is to create a "separation of learning" under $\text{Conv}(p)^{\otimes 4}$, since in the large p limit, learning Y_i requires $d^{(3+i)/4+\epsilon}$ examples/SGD steps.

In the experiments, the width/number of channels is set to 2048/512 for MLPs/CNNs. We sample $m_t = 32 \times 10240$ ($m_v = 10240$) data points randomly from $\boldsymbol{\mathcal{X}}$ as training (test) set with $p = 4$ and $d = 256$. The SGD training configurations (batch size (=10240), learning rate (=1.), momentum (=0.9) etc.) are identical across architectures. To compute the kernels, we rely crucially on *NeuralTangents* (Novak et al., 2020) which is based on *JAX* (Bradbury et al., 2018a).

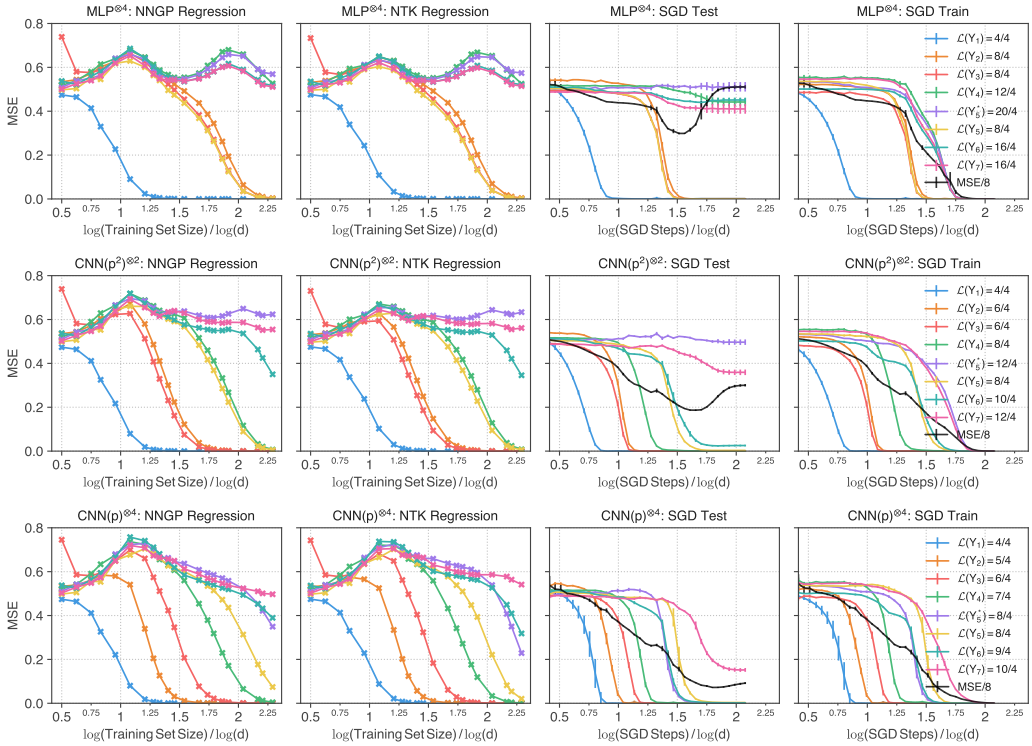


Figure 3: **Learning Dynamics vs Architectures vs Learning Indices.** We plot the learning/training dynamics of each eigenfunction Y_i. From top to bottom: a 4-layer MLP $\text{MLP}^{\otimes 4}$, a 2-layer CNN $\text{Conv}(p^2)^{\otimes 2}$ and a 4-layer CNN $\text{Conv}(p)^{\otimes 4}$. From left to right: residual MSE (per eigenfunction) of NNGP/NTK regression, test/training MSE of SGD. The learning indices of Y_i in each architecture are shown in the legends.

C.2. Experimental Results I: Architectures vs Learnability.

In Fig.3, for each eigenfunction Y_i, we plot $\frac{1}{2}\mathbb{E}|\hat{Y}_i(x,t) - Y_i(x)|_2^2$ against t, where $\hat{Y}_i(x,t)$ is the projection of the prediction onto Y_i and t is either the training steps (SGD) or training set size (kernels). The expectation is taken over the test set. The budget index $r = \log(m_t)/\log(d) \approx 2.28$. As $d = 256$ ($p = 4$) is far from the asymptotic limit, we expect $r = 2.28$ being a *soft* cut-off between learnable and non-learnable indices. Although the theorems assume $d, p \to \infty$, they do

provide good predictions even when d and p are far from ∞. We summarize several key observations below.

1. **MLP**$^{\otimes 4}$ (1st Row.) This architecture can capture all low-frequency interactions ($\deg = 1, 2$, $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$, $\mathbf{Y}_5$) but fail to learn $\deg \geq 3$ interactions, as expected. For MLPs, making the network deeper won't improve its learnability much; see Fig. 7 in the appendix.

2. **Conv**$(p^2)^{\otimes 2}$ (2nd Row.) Learning curves of $\mathbf{Y}_2$/$\mathbf{Y}_3$ are separated from $\mathbf{Y}_5$ because the spatial indices of them are different. Higher-frequency ($\deg = 3, 4$) shorter-range interactions ($\mathbf{Y}_4$, $\mathbf{Y}_6$) become (partially) learnable. Note that $\mathcal{L}(\mathbf{Y}_4) = \frac{8}{4} < r \approx 2.28$ is completely learned by NTK and SGD and $\mathcal{L}(\mathbf{Y}_6) = 2.5 > r \approx 2.28$ is partially learned by NTK and almost completely learned by SGD.

3. **Conv**$(p)^{\otimes 4}$ (3rd Row). We see $\mathcal{L}(Y_i)$ capture the order of learning very/reasonably well in the kernel/SGD setting. To test the ability of Conv$(p)^{\otimes 4}$ in modeling ultra-short-range-ultra-high-frequency interactions, we trace the learning progress of $\mathbf{Y}_5^*$ ($\deg(\mathbf{Y}_5^*) = 5, \mathcal{L}(\mathbf{Y}_5^*) = \frac{8}{4}$.) As expected, while other architectures completely fail to make progress, the NTK/NNGP of Conv$(p)^{\otimes 4}$ makes good progress and the SGD even completes the learning process. Interestingly, $\mathbf{Y}_5$[3] is learned faster than $\mathbf{Y}_5^*$ in the kernel setting but slower in the SGD setting (even slower than $\mathcal{L}(\mathbf{Y}_6) = \frac{9}{4}$), which is unexpected. We suspect it might be due to certain "implicit" effects of SGD. Further investigation is needed to understand it.

C.3. Experimental Results II: GAP vs Flatten.

We compare the SGD learning dynamics of two convolutional architectures: Conv$(p)^{\otimes 3}$-Flatten and Conv$(p)^{\otimes 3}$-GAP. The experimental setup is almost the same as that of Sec. C.1 except the eigenfunctions $\{Y_i\}$ are chosen to be in the RKHS of the NNGP kernel/NTK of Conv$(p)^{\otimes 3}$-GAP and thus of Conv$(p)^{\otimes 3}$-Flatten, i.e., they are shifting-invariant (the invariant group is of order p). Moreover, we still have $\mathscr{L}(Y_i) = (i+3)/4$. For each Y_i, we plot the validation MSE of the residual vs SGD steps in Fig. 4. Overall, the predictions from Theorem 13 and Theorem 14 give excellent agreement with the empirical result. With training set size $m_t = 32 \times 10240$ Fig. 4 (a), the residuals of Y_i for GAP and Flatten are almost indistinguishable from each other for $i \leq 6$ (recall that $\mathscr{L}(\mathbf{Y}_6) = 9/4 = 2.25 < r \approx 2.28$). However, when $i = 7$ the dataset size ($r \approx 2.28$) is relatively small compared to the learning index ($\mathscr{L}(\mathbf{Y}_7) = 2.5$), GAP outperforms Flatten in learning $\mathbf{Y}_7$. In Fig. 4 (b), we increase the training set size by a factor of 4 to $m_t = 4 \times 32 \times 10240 \approx d^{2.53}$. We see that the dynamics of $\mathbf{Y}_7$ between GAP and Flatten become indistinguishable.

To test the robustness of the prediction from Theorem 14 on more practical datasets and models, we perform an additional experiment on ImageNet (Deng et al., 2009) using ResNet (He et al., 2016). We compare the performance of the original ResNet50, denoted by ResNet50-GAP, and a modified version ResNet50-Flatten, in which the GAP readout layer is replaced by Flatten. We use the ImageNet codebase from FLAX[4](Heek et al., 2020). In order to see how the performance difference between ResNet50-GAP and ResNet50-Flatten evolves as the training set size increases, we make a scaling plot, namely, we vary the training set sizes[5] $m_i = [m \times 2^{-i/2}]$ for $i = 0, \ldots, 11$, where $m = 1281167$ is the total number of images in the training set of ImageNet. The networks

3. Ultra-Long-Range-Low-Frequency under Conv$(p)^{\otimes 4}$, with $\deg(\mathbf{Y}_5) = 2$ and $\mathcal{L}(\mathbf{Y}_5) = 8/4 = \mathcal{L}(\mathbf{Y}_5^*)$
4. https://github.com/google/flax/blob/main/examples/imagenet/README.md
5. Standard data-augmentation is applied for each m_i; see input_pipeline.py.

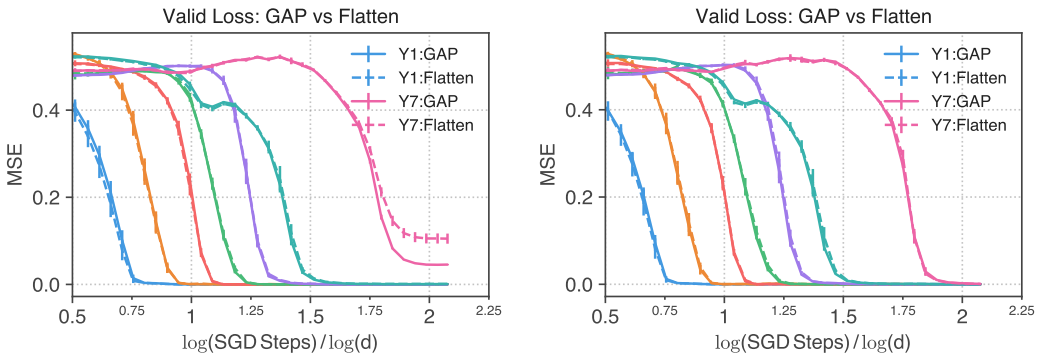
are trained for 150 epochs with batch size 128. We plot the validation accuracy and loss (averaged over 3 runs) as a function of training set size m_i in Fig. 5. Overall, we see that the performance gap between ResNet50-GAP and ResNet50-Flatten shrink substantially as the training set size increases. E.g, using 1/8 of the training set (i.e. $i = 6$), the top 1 accuracy between the two is 19.3% (57.7% GAP vs 38.4% Flatten). However, with the whole training set (i.e., m_0), this gap is reduced to 2% (76.5% GAP vs 74.5% Flatten). To demonstrate the robustness of this trend, we additionally generate the same plots for ResNet34 and ResNet101; see Fig. 8 in the Appendix.

Appendix D. DAGs, Eigenfunctions, Spatial Index, and Frequency Index.

In this section, we provide more details regarding the DAGs and the eigenfunctions used in the experiments, and how the spatial, frequency and learning indices are computed.

Let $(\overline{\mathbb{S}}_{p-1})^{p^3} \subseteq \mathbb{R}^{p^4}$ be the input space, where $d = p^4$ is the input dimension. We use $\boldsymbol{x} \equiv (\boldsymbol{x}_{\boldsymbol{k}})_{\boldsymbol{k}\in[p]^4} \in (\overline{\mathbb{S}}_{p-1})^{p^3}$ to denote one input (an image), where $\boldsymbol{k} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \boldsymbol{k}_4] \in [p]^4$. In addition, we treat $[p]^4$ as a group (i.e. with circular boundaries) and let $\boldsymbol{e}_1 = [1, 0, 0, 0]$, $\boldsymbol{e}_2 = [0, 1, 0, 0]$, $\boldsymbol{e}_3 = [0, 0, 1, 0]$ and $\boldsymbol{e}_4 = [0, 0, 0, 1]$ be a set of generator/basis of the group. Note that each input $\boldsymbol{x}_{\boldsymbol{k}}$ is partitioned into p^3 many patches: $\{\boldsymbol{x}_{\boldsymbol{k}_1,\boldsymbol{k}_2,\boldsymbol{k}_3,:} : \boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3 \in [p]\}$.

The eigenfunctions used in the experiments and the associated space/frequency indices (will be explained momentarily) are given as follows



(*a*) (*b*)

Figure 4: **Learning Dynamics: GAP vs Flatten.** Training Set Size: (a) $m = 32 \times 10240$ (b) $m = 128 \times 10240$. We plot the validation MSE of the residual of each Y_i (**left** $\to$ **right**: $i = 1 \to 7$) for GAP (Solid lines) and Flatten (Dashed lines). The mean/std in each curve is obtained by 5 random initializations. Left: when training set size is 32×10240, the residual dynamics of GAP and Flatten are almost indistinguishable for Y_i with $i \leq 6$. But GAP outperforms Flatten when learning $\mathbf{Y_7}$. Right: the training set size is increased by a factor of 4 and the learning dynamics of $\mathbf{Y_7}$ become identical for GAP and Fatten.

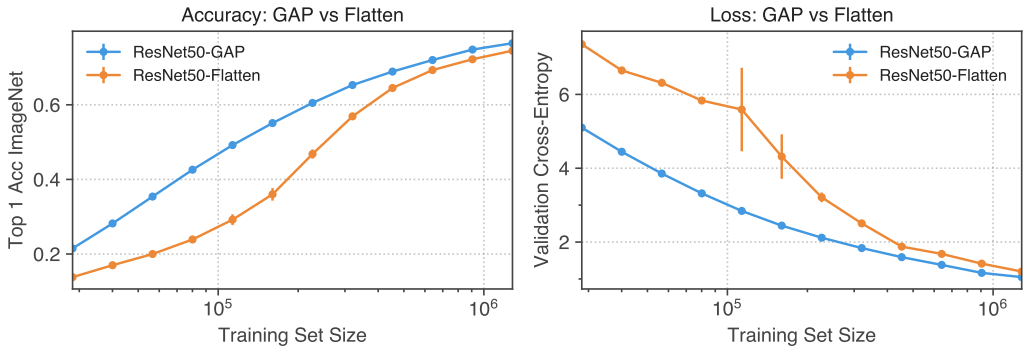


Figure 5: **ResNet50-GAP vs ResNet50-Flatten.** As the training set size increases the performance (accuracy and loss) gap between the two shrinks.

Eigenfunction	**degree**	**Space/Freq Index**		
		MLP	$\text{CNN}(p^2)^{\otimes 2}$	$\text{CNN}(p)^{\otimes 4}$
$\mathrm{Y}_1(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(1)} \boldsymbol{x}_{\boldsymbol{k}}$	1	0/1	$\frac{1}{2}/\frac{1}{2}$	$\frac{3}{4}/\frac{1}{4}$
$\mathrm{Y}_2(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(2)} \boldsymbol{x}_{\boldsymbol{k}}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$	2	0/2	$\frac{1}{2}/\frac{2}{2}$	$\frac{3}{4}/\frac{2}{4}$
$\mathrm{Y}_3(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(3)} \boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_3}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$	2	0/2	$\frac{1}{2}/\frac{2}{2}$	$\frac{4}{4}/\frac{2}{4}$
$\mathrm{Y}_4(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(4)} \boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_3+\boldsymbol{e}_4}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}\boldsymbol{x}_{\boldsymbol{k}}$	3	0/3	$\frac{1}{2}/\frac{3}{2}$	$\frac{4}{4}/\frac{3}{4}$
$\mathrm{Y}_5^*(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(5^*)} \boldsymbol{x}_{\boldsymbol{k}}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}\boldsymbol{x}_{\boldsymbol{k}+2\boldsymbol{e}_4}(\boldsymbol{x}_{\boldsymbol{k}}^2 - \boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}^2)$	5	0/5	$\frac{1}{2}/\frac{5}{2}$	$\frac{3}{4}/\frac{5}{4}$
$\mathrm{Y}_5(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(5)} \boldsymbol{x}_{\boldsymbol{k}}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_1}$	2	0/2	$\frac{2}{2}/\frac{2}{2}$	$\frac{6}{4}/\frac{2}{4}$
$\mathrm{Y}_6(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(6)} \boldsymbol{x}_{\boldsymbol{k}}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_3}(3\boldsymbol{x}_{\boldsymbol{k}-\boldsymbol{e}_3}^2 - \boldsymbol{x}_{\boldsymbol{k}}^2)$	4	0/4	$\frac{1}{2}/\frac{4}{2}$	$\frac{5}{4}/\frac{4}{4}$
$\mathrm{Y}_7(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in[p-1]^4} c_{\boldsymbol{k}}^{(7)} \boldsymbol{x}_{\boldsymbol{k}-\boldsymbol{e}_3+\boldsymbol{e}_4}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_2}(3\boldsymbol{x}_{\boldsymbol{k}}^2 - \boldsymbol{x}_{\boldsymbol{k}-\boldsymbol{e}_3+\boldsymbol{e}_4}^2)$	4	0/4	$\frac{1}{2}/\frac{4}{2}$	$\frac{6}{4}/\frac{4}{4}$

Each (eigen)function Y_i is a linear combination of basis eigenfunctions of the same type, in the sense they have the same eigenvalue and the same spatial/frequency/learning indices. The coefficients $c_{\boldsymbol{k}}^{(i)}$ are first sampled from standard Gaussians iid and then multiplied by an i-dependent constant so that Y_i has unit norm[6]. In the experiments, p is chosen to be 4 and the target is defined to be sum of them

$$Y = \sum Y_i \tag{42}$$

6. In our experiments, they are normalized over the test set.

Since they are orthogonal to each other, $\|Y\|_2^2/8 = \|Y_i\|_2^2 = 1$, where the L^2-norm is taken over the uniform distribution on $(\overline{\mathbb{S}}_{p-1})^{p^3}$. In the experiment when we compare Flatten and GAP (Fig. 4), we set all $c_{\boldsymbol{k}}^{(i)} = c^{(i)}$ for some $c^{(i)}$, so that the functions are learnable by convolutional networks with a GAP readout layer. Note that we also remove $\mathbf{Y_5}$ from the target function Y since $\mathbf{Y_5}$ is not in the function space defined by Conv$(p)^{\otimes 3}$-Flatten or Conv$(p)^{\otimes 3}$-GAP.

We compare three architectures. Fig. 6 Column (a) MLP$^{\otimes 4}$, a (four-layer) MLP, the most coarse architecture used in the paper. Fig. 6 Column (b), CNN$(p^2)^{\otimes 2}$, a"D"-CNN that contains two convolutional layers with filter size/stride equal to p^2. Fig. 6 Column (c), CNN$(p)^{\otimes 4}$, a "HR"-CNN, the finest architecture used in the experiments, that contains four convolutional layers with filter size/stride equal to p. In all experiments except the one in Fig. 4, we use Flatten as the readout layer for the convolutional networks and add a *Act-Dense* layer after Flatten to improve the expressivity of the function class. However, in the Flatten vs GAP experiments, Fig. 4, we have only one dense layer after GAP/Flatten and, in particular, no non-linear activation.

We show how to compute the frequency index, the spatial index and the learning index through three examples: $\mathbf{Y_2}$ / $\mathbf{Y_3}$ / $\mathbf{Y_5^*}$, which have degree 2 / 2 / 5, resp. The indices of other (basis) eigenfunctions can be computed using the same approach. We use **Dashed Lines** to represent either an edge connecting an input node to a node in the first-hidden layer or an edge associated to a dense layer. In either case, the corresponding output node of the edge has degree $O(1)$, and thus, the weights (of the DAGs) of such edges are always 0. Only **Solid Lines** are relevant in computing the *spatial index*. Since each Y_i is a linear combination of basis eigenfunctions of the same type, we only need to compute the indices of one component. For convenience, we compute the $\boldsymbol{k} = 0$ component, of which the associated MST is highlighted with colored lines in each DAG. Recall that $p = 4$, $d = p^4 = 256$ and $m_t = 32 \times 10240 \sim d^{2.28}$ (training set size), i.e. the budget index is roughly $r = 2.28$.

(a.) **MLP** Column (a) Fig. 6. The NTK and NNGP kernels are inner product kernels and the associated DAGs are linked lists. The corresponding DAG has only one input node whose dimension is equal to $d = p^4$. The spatial index is always 0 since the degree of each hidden node is 1 (since $1 = d^0$) and the frequency index is equal to the degree of the eigenfunctions. Thus $\mathscr{L}(\mathbf{Y_2}) = \mathscr{L}(\mathbf{Y_3}) = 2$ and $\mathscr{L}(\mathbf{Y_5^*}) = 5$. Changing the number of layers won't change the learning indices. In sum, learning $\mathbf{Y_2}$ / $\mathbf{Y_3}$ / $\mathbf{Y_5^*}$ using infinite-width MLP requires $d^{\mathbf{2}^+}/d^{\mathbf{2}^+}/d^{\mathbf{5}^+}$ many samples /SGD steps. Clearly, $\mathbf{Y_5^*}$ is completely unlearnable as $r = 2.28 << 5$. In the MSE plot $\mathbf{Y_5^*}$ (5-th row in Fig. 6), the **Red Lines** does not make any progress.

(b.) **CNN**$(p^2)^{\otimes 2}$ Column (b) Fig. 6. The input image is partitioned into p^2 patches and each patch has dimension p^2. The second layer of the DAG has p^2 many nodes, each node represents one pixel (with many channels) in the first hidden layer of a finite-width ConvNet. After one more convolutional layer with filter size/stride p^2, the number of node (pixel) is reduced to one. The remaining part of the DAG is essentially a linked list (**Dashed Line**) with length equal to 1, which corresponds to the *Act-Dense* layer. The frequency index $\mathscr{F}(\mathbf{Y_2}) = 2\frac{1}{2} = 1$. This is because the degree of $\mathbf{Y_2}$ is 2 and the input dimension of a node is $p^2 = d^{1/2}$. The spatial index is equal to $1/2$, since the minimum tree containing $\boldsymbol{x}_{\boldsymbol{k}}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ has only one non-zero edge (**Solid Lines**) whose weight is equal to 1/2 (since the degree of the output node is $p^2 = d^{1/2}$); see the **colored paths** in Fig. 6 Column (b). Therefore the learning index of

$\mathcal{L}(\mathbf{Y_2}) = 1 + 1/2 = 3/2$. Similarly $\mathcal{L}(\mathbf{Y_3}) = 1 + 1/2 = 3/2$, as the term $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_3}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ are lying in the same patch of size p^2 for all $\boldsymbol{k}$, and $\mathcal{L}(\mathbf{Y}_5^*) = 5/2 + 1/2 = 3$. In sum, learning $\mathbf{Y_2}$ / $\mathbf{Y_3}$ / $\mathbf{Y}_5^*$ using infinite-width CNN$(p^2)^{\otimes 2}$ requires $d^{1.5^+}/d^{1.5^+}/d^{3^+}$ many samples / SGD steps. While neither infinite-width CNN$(p^2)^{\otimes 2}$ nor MLP$^{\otimes 4}$ distinguishes $\mathbf{Y_2}$ from $\mathbf{Y_3}$, CNN$(p^2)^{\otimes 2}$ does improve the learning efficiency for both of them: $d^{2^+} \to d^{1.5^+}$. Note that $\mathbf{Y}_5^*$ is still unlearnable as $r = 2.28 < 3 = \mathcal{L}(\mathbf{Y}_5^*)$. In the MSE plot $\mathbf{Y}_5^*$ (5-th row in Fig. 6), the **Orange Line** does not make any progress. This is also the case for finite-width network trained by SGD; see second row in Fig. 3.

(c.) **CNN**$(p)^{\otimes 4}$ Column (c) Fig. 6. The input image is partitioned into p^3 patches and each patch has dimension p. The second/third/fourth/output layer of the DAG has $p^3/p^2/p/1$ many nodes. The frequency indices are: $\mathcal{F}(\mathbf{Y_2}) = \mathcal{F}(\mathbf{Y_3}) = 2\frac{1}{4} = 1/2$ and $\mathcal{F}(\mathbf{Y}_5^*) = 5\frac{1}{4} = 5/4$. This is because the size of input nodes is reduced to $p = d^{1/4}$. Unlike the above cases, the spatial indices become different. The two interacting terms in $\boldsymbol{x}_{\boldsymbol{k}}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ and the three interacting terms in $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}\boldsymbol{x}_{\boldsymbol{k}+2\boldsymbol{e}_4}(\boldsymbol{x}_{\boldsymbol{k}}^2 - \boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}^2)$ are in the same input node while the two interacting terms in $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_3}\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ and are in two different input nodes. As a consequence, the minimum spanning tree (MST) that contains $\boldsymbol{x}_{\boldsymbol{k}}$ and $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ and the one contains $\boldsymbol{x}_{\boldsymbol{k}}$, $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ and $\boldsymbol{x}_{\boldsymbol{k}+2\boldsymbol{e}_4}$ are the same. They have **3 solid lines**. However, the MST containing $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_3}$ and $\boldsymbol{x}_{\boldsymbol{k}+\boldsymbol{e}_4}$ has **4 solid lines**. Therefore $\mathcal{S}(\mathbf{Y_2}) = \mathcal{S}(\mathbf{Y}_5^*) = 3 \times \frac{1}{4}$ and $\mathcal{S}(\mathbf{Y_3}) = 4 \times \frac{1}{4}$. As such, $\mathcal{L}(\mathbf{Y_2}) = \frac{5}{4}$, $\mathcal{L}(\mathbf{Y_3}) = \frac{6}{4}$ and $\mathcal{L}(\mathbf{Y}_5^*) = \frac{8}{4}$. In sum, learning $\mathbf{Y_2}$ / $\mathbf{Y_3}$ / $\mathbf{Y}_5^*$ using infinite-width CNN$(p)^{\otimes 4}$ requires $d^{1.25^+}$ / $d^{1.5^+}$ / d^{2^+} many samples / SGD steps, resp. Now $\mathcal{L}(\mathbf{Y}_5^*) = 2. < 2.28$ and in the MSE plot $\mathbf{Y}_5^*$ (5-th row in Fig. 6), the **Blue Line** does make significant progress (the MSE is reduced from ~ 0.5 to ~ 0.2.) Finite-width network trained by SGD does even better: the test MSE is almost zero; see third row in Fig. 3.

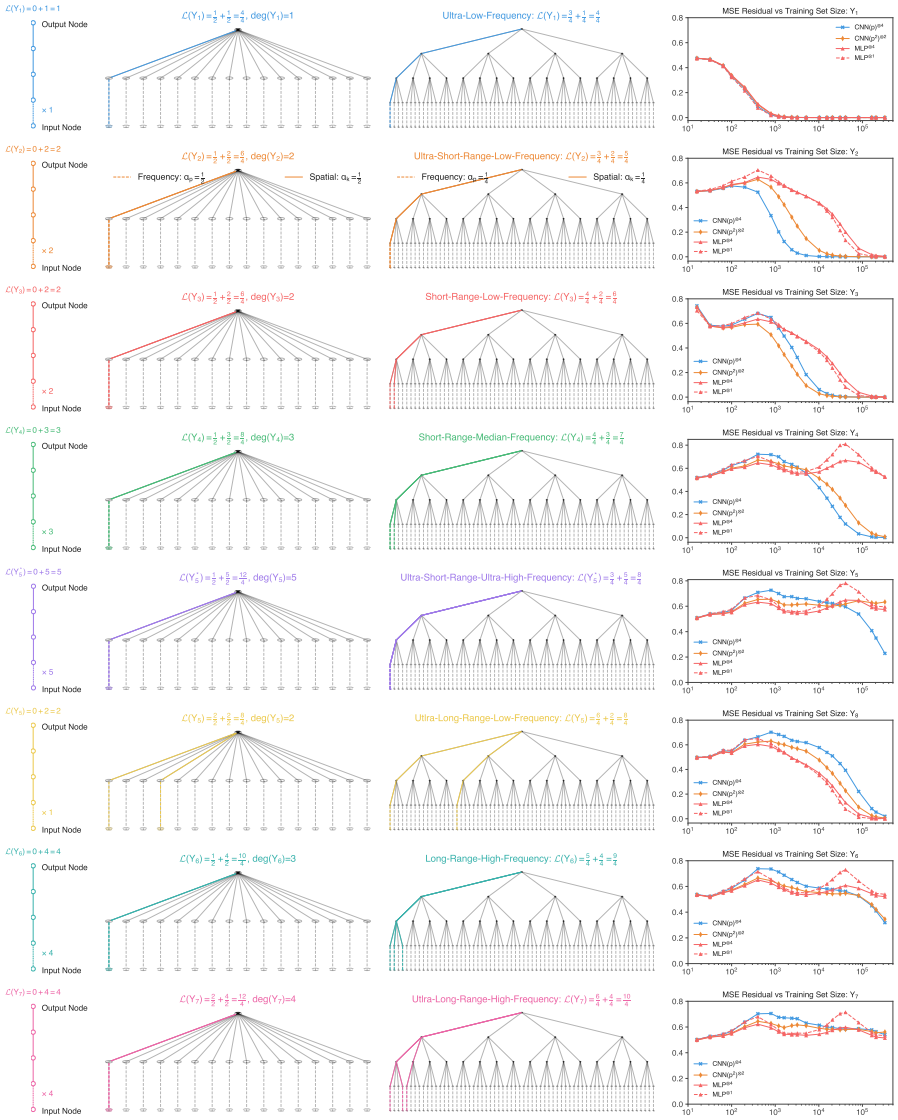


Figure 6: **Eigenfunction vs Learning Index vs Architecture/DAG.** Rows: eigenfunctions Y_i with various space-frequency combinations. Columns: DAGs associated to (a) CNN$(p)^{\otimes 4}$, a "HR"-CNN. (b) CNN$(p^2)^{\otimes 2}$, a"D"-CNN. (c) a four-layer MLP. Column (d) is the MSE of the residual of the corresponding eigenfunction obtained by NTK-regression. The dashed lines in each DAG correspond to the mapping from the input layer to the first hidden layer and the associated weights to the DAG is ZERO. Each colored path in each DAG corresponds to the minimum spanning tree that contains all interaction terms of the corresponding eigenfunctions.

Appendix E. Proof of the Eigenspace Restructuring Theorem.

The goal of this section is to prove the eigenspace restructuring theorem. In Sec. E.1, we present a key lemma that relates the mixed derivatives of $\mathcal{K}$ and Θ (as a function of $\boldsymbol{t}$) to the architectures of the networks. We briefly recap some tools from spherical harmonics and then prove the theorem in

Sec.E.2. For the rest of the paper, we will use the following notations. For $A, B : \mathbb{N} \to \mathbb{R}^+$,

$$B(d) \gtrsim A(d) \iff A(d) \lesssim B(d) \iff \exists c,\, d_0 > 0 \quad s.t. \quad B(d) \geq cA(d) > 0 \quad \text{for all} \quad d > d_0 \tag{43}$$

and

$$B(d) \sim A(d) \iff B(d) \gtrsim A(d) \quad \text{and} \quad A(d) \gtrsim B(d) \tag{44}$$

E.1. A Crucial Lemma

Lemma 6 *Same assumptions as Theorem 3. Let $\boldsymbol{r} \in \mathbb{N}^{|\mathcal{N}_0^{(d)}|}$. Then for $\boldsymbol{r} \neq 0$, for d large enough,*

$$\mathcal{K}_{\mathcal{G}}^{(\boldsymbol{r})}(0), \quad \Theta_{\mathcal{G}}^{(\boldsymbol{r})}(0) \sim d^{-\mathbb{S}(\boldsymbol{r})} \quad \textit{and} \quad \|\mathcal{K}_{\mathcal{G}}^{(\boldsymbol{r})}\|_\infty,\ \|\Theta_{\mathcal{G}}^{(\boldsymbol{r})}\|_\infty \lesssim d^{-\mathbb{S}(\boldsymbol{r})} \tag{45}$$

Proof Recall that the recursion formulas for $\mathcal{K}$ and Θ are

$$\mathcal{K}_u(\boldsymbol{t}) = \phi_u^*\left(\fint_{v:uv\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t})\right) \tag{46}$$

$$\Theta_u(\boldsymbol{t}) = \dot{\phi}_u^*\left(\fint_{v:uv\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t})\right) \fint_{v:uv\in\mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \tag{47}$$

For convenience, define $\overline{\mathcal{K}}$, $\overline{\Theta}$, $\dot{\mathcal{K}}$ as follows

$$\overline{\mathcal{K}}_u(\boldsymbol{t}) = \fint_{v:uv\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) \tag{48}$$

$$\overline{\Theta}_u(\boldsymbol{t}) = \fint_{v:uv\in\mathcal{E}} \Theta_v(\boldsymbol{t}) \tag{49}$$

$$\dot{\mathcal{K}}_u(\boldsymbol{t}) = \dot{\phi}_u^* \circ \overline{\mathcal{K}}_u(\boldsymbol{t})\,. \tag{50}$$

Note that

$$\mathcal{K}_u(\boldsymbol{t}) = \phi_u^* \circ \overline{\mathcal{K}}_u(\boldsymbol{t}) \quad \text{and} \quad \Theta(\boldsymbol{t}) = \dot{\mathcal{K}}_u(\boldsymbol{t})(\overline{\mathcal{K}}_u(\boldsymbol{t}) + \overline{\Theta}_u(\boldsymbol{t}))\,.$$

and $\mathcal{K}_u(\boldsymbol{0}) = \Theta_u(\boldsymbol{0}) = 0$ if $u \neq o_{\mathcal{G}}$, which follow directly from recursions and the fact $\phi_u^*(0) = 0$ for $u \in \mathcal{N}^{(d)}$ if $u \neq o_{\mathcal{G}}$.

We induct on the tuple $(h, |\boldsymbol{r}|)$, where $h \geq 0$ is the number of *hidden* layers in $\mathcal{G}^{(d)}$ and $|\boldsymbol{r}|$ is the total degree of $\boldsymbol{r}$. We begin with the proof of the NNGP kernel $\mathcal{K}$.

Base Case I: $|\boldsymbol{r}| = 1$ and $h \geq 0$ is any integer. Let $u \in \mathcal{N}_0$ be such that $\boldsymbol{r} = \boldsymbol{e}_u$, where $\{\boldsymbol{e}_u\}_{u\in\mathcal{N}_0^{(d)}}$ is the standard basis. Then

$$\partial_{\boldsymbol{t}_u}\mathcal{K}_{\mathcal{G}}(\boldsymbol{t}) = \sum_{\text{path}\in\mathcal{P}(u\to o_{\mathcal{G}})} \prod_{v\in\text{path}} \deg(v)^{-1}\dot{\phi}_v^* \circ \overline{\mathcal{K}}_v(\boldsymbol{t}) \sim \sum_{\text{path}\in\mathcal{P}(u\to o_{\mathcal{G}})} \prod_{v\in\text{path}} d^{-\alpha_v}\dot{\phi}_v^* \circ \overline{\mathcal{K}}_v(\boldsymbol{t}) \tag{51}$$

Here $\mathcal{P}(u \to u')$ represents the set of paths from u to u'. By **Assumption-$\mathcal{G}$** and **Assumption-ϕ**, $|\mathcal{P}(u \to o_{\mathcal{G}})|$ (the cardinality) is uniformly bounded and $\dot{\phi}_v^*(0) > 0$ for all hidden nodes v. Therefore

$$\begin{aligned}\partial_{\boldsymbol{t}_u}\mathcal{K}_{\mathcal{G}}(\mathbf{0}) &\sim \sum_{\text{path}\in\mathcal{P}(u\to o_{\mathcal{G}})} d^{-\sum_{v\in\text{path}}\alpha_v}\dot{\phi}_v^*\circ\overline{\mathcal{K}}_v(\mathbf{0})\\ &= \sum_{\text{path}\in\mathcal{P}(u\to o_{\mathcal{G}})} d^{-\sum_{v\in\text{path}}\alpha_v}\dot{\phi}_v^*(0)\\ &\sim \max_{\text{path}\in\mathcal{P}(u\to o_{\mathcal{G}})} d^{-\sum_{v\in\text{path}}\alpha_v}\\ &= d^{-\mathcal{S}(\boldsymbol{e}_u)} = d^{-\mathcal{S}(\boldsymbol{r})}\end{aligned}$$

In the above, we have used $\overline{\mathcal{K}}_v(\mathbf{0}) = 0$ (which is due to $\phi^*(0) = 0$) and $\dot{\phi}_v^*\circ\overline{\mathcal{K}}_v(\mathbf{0}) \neq 0$. The second estimate $\|\partial_{t_u}\mathcal{K}_{\mathcal{G}}\|_\infty \lesssim d^{-\mathcal{S}(\boldsymbol{r})}$ follows from $|\dot{\phi}_v^*\circ\overline{\mathcal{K}}_v(\boldsymbol{t})| \lesssim 1$.

Base Case II: $h = 0$ and $|\boldsymbol{r}| \geq 1$ is any number. Note that $\mathcal{G}^{(d)}$ has no hidden layer and all input nodes are linked to the output node $o_{\mathcal{G}}$. The case when the activation $\phi_{o_{\mathcal{G}}}$ is the identity function is obvious. We assume $\phi_{o_{\mathcal{G}}}$ is semi-admissible.

$$\partial_{\boldsymbol{t}}^{\boldsymbol{r}}\mathcal{K}_{\mathcal{G}}(\boldsymbol{t}) = \deg(o_{\mathcal{G}})^{-|\boldsymbol{r}|}{\phi_{o_{\mathcal{G}}}^*}^{(|\boldsymbol{r}|)}\left(\fint_{u\in\mathcal{N}_0^{(d)}}\boldsymbol{t}_u\right).$$

This implies Eq. (45) since $\mathcal{S}(\boldsymbol{r}) = 0$, $\deg(o_{\mathcal{G}}) \lesssim 1$ by **Assumption-$\mathcal{G}$** and ${\phi_{o_{\mathcal{G}}}^*}^{(|\boldsymbol{r}|)}(0) > 0$ by **Assumption-ϕ**.

Induction: $|\boldsymbol{r}| \geq 2$, $h \geq 1$ and $\boldsymbol{r} \in \mathcal{A}(\mathcal{G}^{(d)})$. We only prove the first estimate in Eq. (45) since the other one can be proved similarly. WLOG, we assume $\phi_{o_{\mathcal{G}}}$ is not the identity function and hence is semi-admissible. Let $u \in \mathcal{N}_0^{(d)}$ be such that $\boldsymbol{r}_u \geq 1$ and denote $\bar{\boldsymbol{r}} = \boldsymbol{r} - \boldsymbol{e}_u$. Then

$$\partial_{\boldsymbol{t}}^{\boldsymbol{r}}\mathcal{K}_{\mathcal{G}}(\boldsymbol{t})\Big|_{\boldsymbol{t}=\mathbf{0}} = \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}}(\partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_{\mathcal{G}}(\boldsymbol{t}))\Big|_{\boldsymbol{t}=\mathbf{0}} = \sum_{\substack{o_{\mathcal{G}}v\in\mathcal{E}\\ \partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_v\neq 0}} \deg(o_{\mathcal{G}})^{-1}\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}}\left(\dot{\mathcal{K}}_{o_{\mathcal{G}}}(\boldsymbol{t})\partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_v(\boldsymbol{t})\right)\Big|_{\boldsymbol{t}=\mathbf{0}} \tag{52}$$

$$= \sum_{\substack{o_{\mathcal{G}}v\in\mathcal{E}\\ \partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_v\neq 0}} \sum_{\bar{\boldsymbol{r}}_1+\bar{\boldsymbol{r}}_2=\bar{\boldsymbol{r}}} \deg(o_{\mathcal{G}})^{-1}\left(\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_1}\dot{\mathcal{K}}_{o_{\mathcal{G}}}(\boldsymbol{t})\,\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2+\boldsymbol{e}_u}\mathcal{K}_v(\boldsymbol{t})\right)\Big|_{\boldsymbol{t}=\mathbf{0}} \tag{53}$$

$$\sim \sum_{\substack{o_{\mathcal{G}}v\in\mathcal{E}\\ \partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_v\neq 0}} \sum_{\bar{\boldsymbol{r}}_1+\bar{\boldsymbol{r}}_2=\bar{\boldsymbol{r}}} \deg(o_{\mathcal{G}})^{-1}d^{-\mathcal{S}(\bar{\boldsymbol{r}}_1)}d^{-\mathcal{S}(n(\bar{\boldsymbol{r}}_2+\boldsymbol{e}_u;v))} \tag{54}$$

$$\sim \sum_{\substack{o_{\mathcal{G}}v\in\mathcal{E}\\ \partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_v\neq 0}} \sum_{\bar{\boldsymbol{r}}_1+\bar{\boldsymbol{r}}_2=\bar{\boldsymbol{r}}} d^{-\mathcal{S}(\bar{\boldsymbol{r}}_1)}d^{-\left(\alpha_{o_{\mathcal{G}}}+\mathcal{S}(n(\bar{\boldsymbol{r}}_2+\boldsymbol{e}_u;v))\right)} \tag{55}$$

$$\sim \sup_{\substack{o_{\mathcal{G}}v\in\mathcal{E}\\ \partial_{\boldsymbol{t}}^{\boldsymbol{e}_u}\mathcal{K}_v\neq 0}} \sup_{\bar{\boldsymbol{r}}_1+\bar{\boldsymbol{r}}_2=\bar{\boldsymbol{r}}} d^{-\left(\mathcal{S}(\bar{\boldsymbol{r}}_1)+\alpha_{o_{\mathcal{G}}}+\mathcal{S}(n(\bar{\boldsymbol{r}}_2+\boldsymbol{e}_u;v))\right)} \tag{56}$$

We have applied induction twice in Eq. (54): one to obtain the estimate $\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_1}\dot{\mathcal{K}}_{o_{\mathcal{G}}}^*(\mathbf{0}) \sim d^{-\mathcal{S}(\bar{\boldsymbol{r}}_1)}$ (with $|\bar{\boldsymbol{r}}_1| < |\boldsymbol{r}|$ and $\dot{\phi}_{o_{\mathcal{G}}}^*$ semi-admissible) and one to $\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2+\boldsymbol{e}_u}\mathcal{K}_v(\boldsymbol{t}) \sim d^{-\mathcal{S}(n(\bar{\boldsymbol{r}}_2+\boldsymbol{e}_u;v))}$, in which the subgraph with v as the output node has depth at most $(h-1)$. The last line follows from that both the cardinality of the tuple $(\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_2)$ with $\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_2 \geq \mathbf{0}$ and $\bar{\boldsymbol{r}}_1 + \bar{\boldsymbol{r}}_2 = \bar{\boldsymbol{r}}$ and the cardinality of $v \in \mathcal{N}_0^{(d)}$

with $o_{\mathcal{G}} v \in \mathcal{E}$ and $\partial_{\boldsymbol{t}}^{\boldsymbol{e}_u} \mathcal{K}_v \neq 0$ are finite and independent of d. From the definition of MST, it is clear that for all $(\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_2)$

$$\mathcal{S}(\bar{\boldsymbol{r}}_1) + \alpha_{o_{\mathcal{G}}} + \mathcal{S}(\boldsymbol{n}(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u; v)) \geq \mathcal{S}(\bar{\boldsymbol{r}}_1) + \mathcal{S}(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u) \geq \mathcal{S}(\boldsymbol{r}) \tag{57}$$

It remains to show that there exists at least one pair $(\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_2)$ such that the above can be an equality. Let $\mathcal{T} \subseteq \mathcal{G}^{(d)}$ be a MST containing all nodes in $\boldsymbol{n}(\boldsymbol{r})$. If $o_{\mathcal{G}}$ has only one child v in $\mathcal{T}$, then we choose $\bar{\boldsymbol{r}}_1 = 0$ and notice that

$$\mathcal{S}(\bar{\boldsymbol{r}}_1) + \alpha_{o_{\mathcal{G}}} + \mathcal{S}(\boldsymbol{n}(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u; v)) = 0 + \mathcal{S}(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u) = \mathcal{S}(\boldsymbol{r}) \tag{58}$$

since $\mathcal{S}(\bar{\boldsymbol{r}}_1) = 0$ and $\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u = \boldsymbol{r}$. Else, $o_{\mathcal{G}}$ contains at least two children in $\mathcal{T}$ and therefore at least two disjoint branches. Let $\mathcal{T}_u \subseteq \mathcal{T}$ be the branch that contains u and choose $\bar{\boldsymbol{r}}_2 \leq \bar{\boldsymbol{r}}$ be such that all the nodes of $(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u)$ are contained in $\mathcal{T}_u$ and all the nodes of $\bar{\boldsymbol{r}}_1 \equiv \boldsymbol{r} - (\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u)$ are contained in $\mathcal{T} \backslash \mathcal{T}_u$. Clearly

$$\mathcal{S}(\boldsymbol{r}) = \mathcal{S}(\bar{\boldsymbol{r}}_1) + \mathcal{S}(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u) = \mathcal{S}(\bar{\boldsymbol{r}}_1) + \mathcal{S}(\boldsymbol{n}(\bar{\boldsymbol{r}}_2 + \boldsymbol{e}_u; v)) + \alpha_{o_{\mathcal{G}}}, \tag{59}$$

where v is the unique child of $o_{\mathcal{G}}$ in $\mathcal{T}_u$.

This completes the proof of the NNGP kernel $\mathcal{K}$. As the proof of the NTK part is quite similar, we will be brief and focus only on the induction step.

Induction Step of Θ: $|\boldsymbol{r}| \geq 2$, $h \geq 1$ and $\boldsymbol{r} \in \mathcal{A}(\mathcal{G}^{(d)})$. Recall that the formula of Θ is

$$\Theta_u(\boldsymbol{t}) = \dot{\phi}_u^* \left(\fint_{v: uv \in \mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) \right) \fint_{v: uv \in \mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \tag{60}$$

For $\boldsymbol{r} \in \mathbb{N}^{|\mathcal{N}_0^{(d)}|}$,

$$\partial_{\boldsymbol{t}}^{\boldsymbol{r}} \Theta_{o_{\mathcal{G}}}(\boldsymbol{t}) = \sum_{\bar{\boldsymbol{r}}_1 + \bar{\boldsymbol{r}}_2 = \boldsymbol{r}} \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_1} \dot{\phi}_{o_{\mathcal{G}}}^* \left(\fint_{v: o_{\mathcal{G}} v \in \mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) \right) \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2} \fint_{v: o_{\mathcal{G}} v \in \mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \tag{61}$$

Note that $\dot{\phi}_{o_{\mathcal{G}}}^*$ is semi-admissible. We apply the result of $\mathcal{K}$ to conclude that

$$\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_1} \dot{\phi}_{o_{\mathcal{G}}}^* \left(\fint_{v: o_{\mathcal{G}} v \in \mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) \right) \Bigg|_{\boldsymbol{t} = \boldsymbol{0}} \sim d^{-\mathcal{S}(\boldsymbol{r}_1)} \tag{62}$$

$$\left\| \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_1} \dot{\phi}_{o_{\mathcal{G}}}^* \left(\fint_{v: o_{\mathcal{G}} v \in \mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) \right) \right\|_{\infty} \lesssim d^{-\mathcal{S}(\bar{\boldsymbol{r}}_1)} \tag{63}$$

and the inductive step to conclude that

$$\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2} \fint_{v: o_{\mathcal{G}} v \in \mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \Bigg|_{\boldsymbol{t} = \boldsymbol{0}} \sim \sum_{\substack{v: o_{\mathcal{G}} v \in \mathcal{E} \\ \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2} (\mathcal{K}_v + \Theta_v) \not\equiv \boldsymbol{0}}} d^{-\mathcal{S}(\bar{\boldsymbol{r}}_2)} \quad \text{if} \quad \bar{\boldsymbol{r}}_2 \neq \boldsymbol{0} \quad \text{else} \quad 0 \tag{64}$$

$$\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2} \fint_{v: o_{\mathcal{G}} v \in \mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \lesssim \sum_{\substack{v: o_{\mathcal{G}} v \in \mathcal{E} \\ \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2} (\mathcal{K}_v + \Theta_v) \not\equiv \boldsymbol{0}}} d^{-\mathcal{S}(\bar{\boldsymbol{r}}_2)} . \tag{65}$$

Note that

$$|\{v : o_{\mathcal{G}} v \in \mathcal{E}^{(d)} \quad \text{and} \quad \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2}(\mathcal{K}_v + \Theta_v) \not\equiv \mathbf{0}\}| \lesssim 1\,.$$

Thus

$$\left\|\partial_{\boldsymbol{t}}^{\boldsymbol{r}} \Theta_{o_{\mathcal{G}}}(\boldsymbol{t})\right\|_\infty \lesssim \sum_{\bar{\boldsymbol{r}}_1 + \bar{\boldsymbol{r}}_2 = \boldsymbol{r}} d^{-\mathbb{S}(\bar{\boldsymbol{r}}_1) - \mathbb{S}(\bar{\boldsymbol{r}}_2)} \lesssim d^{-\mathbb{S}(\boldsymbol{r})}. \tag{66}$$

To control the lower bound, let $\mathcal{T}$ be a MST containing $n(\boldsymbol{r})$. If $\deg(o_{\mathcal{G}}; \mathcal{T}) = 1$, then we can choose $\bar{\boldsymbol{r}}_1 = 0$ and $\bar{\boldsymbol{r}}_2 = \boldsymbol{r} \neq \mathbf{0}$. Notice that there is at least one child node v of $o_{\mathcal{G}}$ with $\partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2}(\mathcal{K}_v + \Theta_v) \not\equiv \mathbf{0}$. Therefore

$$\sum_{v : \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2}(\mathcal{K}_v + \Theta_v) \not\equiv \mathbf{0}} d^{-\mathbb{S}(\bar{\boldsymbol{r}}_2)} \gtrsim d^{-\mathbb{S}(\bar{\boldsymbol{r}}_2)} = d^{-\mathbb{S}(\boldsymbol{r})} \tag{67}$$

Combining with

$$\dot{\phi}^*_{o_{\mathcal{G}}} \left(\fint_{v : o_{\mathcal{G}} v \in \mathcal{E}} \mathcal{K}_v(\mathbf{0}) \right) = \dot{\phi}^*_{o_{\mathcal{G}}}(0) > 0 \tag{68}$$

we have

$$\partial_{\boldsymbol{t}}^{\boldsymbol{r}} \Theta_{o_{\mathcal{G}}}(\mathbf{0}) \gtrsim d^{-\mathbb{S}(\boldsymbol{r})}$$

It remains to handle the $\deg(o_{\mathcal{G}}; \mathcal{T}) > 1$ case. We choose $(\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_2)$ such that one branch of $\mathcal{T}$ is the MST that contains $n(\bar{\boldsymbol{r}}_2)$ and $o_{\mathcal{G}}$, and the remaining branch(es) is a MST that contains $n(\bar{\boldsymbol{r}}_1)$ and $o_{\mathcal{G}}$. Then

$$\begin{aligned} \partial_{\boldsymbol{t}}^{\boldsymbol{r}} \Theta_{o_{\mathcal{G}}}(\mathbf{0}) \gtrsim & \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_1} \dot{\phi}^*_{o_{\mathcal{G}}} \left(\fint_{v : o_{\mathcal{G}} v \in \mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) \right) \partial_{\boldsymbol{t}}^{\bar{\boldsymbol{r}}_2} \fint_{v : o_{\mathcal{G}} v \in \mathcal{E}} (\mathcal{K}_v(\boldsymbol{t}) + \Theta_v(\boldsymbol{t})) \Big|_{\boldsymbol{t}=\mathbf{0}} \\ \gtrsim & d^{-\mathbb{S}(\boldsymbol{r}_1) - \mathbb{S}(\boldsymbol{r}_2)} = d^{-\mathbb{S}(\boldsymbol{r})} \end{aligned}$$

■

E.2. Legendre Polynomials, Spherical Harmonics and their Tensor Products.

Our notation follows closely from (Frye and Efthimiou, 2012).

Legendre Polynomials. Let $d_{\text{in}} \in \mathbb{N}^*$ and $\omega_{d_{\text{in}}}$ be the measure defined on the interval $I = [-1, 1]$

$$\omega_{d_{\text{in}}}(t) = (1 - t^2)^{(d_{\text{in}} - 3)/2} \tag{69}$$

The Legendre polynomials[7] $\{P_r(t) : r \in \mathbb{N}\}$ is an orthogonal basis for the Hilbert space $L^2(I, \omega_{d_{\text{in}}})$, i.e.

$$\int_I P_r(t) P_{r'}(t) \omega_{d_{\text{in}}}(t) dt = 0 \quad \text{if} \quad r \neq r' \quad \text{else} \quad N(d_{\text{in}}, r)^{-1} \left(\frac{|\mathbb{S}_{d_{\text{in}}-1}|}{|\mathbb{S}_{d_{\text{in}}-2}|} \right) \tag{70}$$

Here $P_r(t)$ is a degree r polynomials with $P_r(1) = 1$ that satisfies the formula below, $N(d_{\text{in}}, r)$ is the cardinality of degree r spherical harmonics in $\mathbb{R}^{d_{\text{in}}}$ and $|\mathbb{S}_{d_{\text{in}}-1}|$ is the surface area of $\mathbb{S}_{d_{\text{in}}-1}$.

7. More accurate, this should be called Gegenbauer Polynomials. However, we stick to the terminology in (Frye and Efthimiou, 2012)

Lemma 7 (Rodrigues Formula. Proposition 4.19 (Frye and Efthimiou, 2012))

$$P_r(t) = c_r \omega_{d_{in}}^{-1}(t) \left(\frac{d}{dt}\right)^r (1-t^2)^{r+(d_{in}-3)/2}, \tag{71}$$

where

$$c_r = \frac{(-1)^r}{2^r (r+(d_{in}-3)/2)_r} \tag{72}$$

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

$$(x)_l \equiv x(x-1)\cdots(x-l+1) \tag{73}$$
$$(x)_0 \equiv 1 \tag{74}$$

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

$$|\mathbb{S}_{d_{\text{in}}-1}| \equiv \int_{\mathbb{S}_{d_{\text{in}}-1}} d\mathbb{S}_{d_{\text{in}}-1} = \frac{2\pi^{d_{\text{in}}/2}}{\Gamma(\frac{d_{\text{in}}}{2})}. \tag{75}$$

The normalized measure on this sphere is defined to be

$$d\sigma_{d_{\text{in}}} = \frac{1}{|\mathbb{S}_{d_{\text{in}}-1}|} d\mathbb{S}_{d_{\text{in}}-1} \quad \text{and} \quad \int_{\mathbb{S}_{d_{\text{in}}-1}} d\sigma_{d_{\text{in}}} = 1. \tag{76}$$

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

$$\int_{\xi \in \mathbb{S}_{d_{\text{in}}-1}} Y_{r,l}(\xi) Y_{r',l'}(\xi) d\sigma_{d_{\text{in}}} = \delta_{(r,l)=(r',l')}. \tag{77}$$

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

$$N(d_{\text{in}}, r) = \frac{2r+d_{\text{in}}-2}{r} \binom{d_{\text{in}}+r-3}{r-1} \sim (d_{\text{in}})^r / r! \quad \text{as} \quad d_{\text{in}} \to \infty. \tag{78}$$

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

Lemma 8 (Addition Theorem. Theorem 4.11 (Frye and Efthimiou, 2012))

$$P_r(\xi^T \eta) = \frac{1}{N(d_{in}, r)} \sum_{l \in [N(d_{in}, r)]} Y_{r,l}(\xi) Y_{r,l}(\eta), \quad \xi, \eta \in \mathbb{S}_{d_{in}-1}. \tag{79}$$

Tensor Products. Let $\boldsymbol{d} = (\boldsymbol{d}_u)_{u \in \mathcal{N}_0^{(d)}} \in \mathbb{N}^{|\mathcal{N}_0^{(d)}|}$, $\boldsymbol{r} \in \mathbb{N}^{|\mathcal{N}_0^{(d)}|}$, $\boldsymbol{I} = I^{|\mathcal{N}_0^{(d)}|} = [-1,1]^{|\mathcal{N}_0^{(d)}|}$ and $\boldsymbol{\omega} = \bigotimes_{u \in \mathcal{N}_0^{(d)}} \omega_{\boldsymbol{d}_u}$ be a product measure on $\boldsymbol{I}$. Then the (product of) Legendre polynomials

$$\boldsymbol{P}_{\boldsymbol{r}}(\boldsymbol{t}) = \prod_{u \in \mathcal{N}_0^{(d)}} P_{\boldsymbol{r}_u}(\boldsymbol{t}_u), \quad \boldsymbol{t} = (\boldsymbol{t}_u)_{u \in \mathcal{N}_0^{(d)}} \in \boldsymbol{I}, \tag{80}$$

which form an orthogonal basis for the Hilbert space $L^2(\boldsymbol{I},\boldsymbol{\omega}) = \bigotimes_{u\in\mathcal{N}_0^{(d)}} L^2(I,\omega_{\boldsymbol{d}_u})$. Similarly, the tensor product of spherical harmonics

$$\boldsymbol{Y}_{\boldsymbol{r},\boldsymbol{l}} = \prod_{u\in\mathcal{N}_0^{(d)}} Y_{\boldsymbol{r}_u,\boldsymbol{l}_u}, \quad \boldsymbol{l} = (\boldsymbol{l}_u)_{u\in\mathcal{N}_0^{(d)}} \in [\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})] \equiv \prod_{u\in\mathcal{N}_0^{(d)}} [N(\boldsymbol{d}_u,\boldsymbol{r}_u)] \tag{81}$$

form an orthonormal basis for the product space

$$L^2(\boldsymbol{\mathcal{X}},\boldsymbol{\sigma}) \equiv \bigotimes_{u\in\mathcal{N}_0^{(d)}} L^2(\mathbb{S}_{\boldsymbol{d}_u-1},\sigma_{\boldsymbol{d}_u}) \tag{82}$$

Elements in the set $\{\boldsymbol{Y}_{\boldsymbol{r},\boldsymbol{l}}\}_{\boldsymbol{l}\in[\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})]}$ are called degree (order) $\boldsymbol{r}$ spherical harmonics in $L^2(\boldsymbol{\mathcal{X}},\boldsymbol{\sigma})$ and also degree r spherical harmonics if $|\boldsymbol{r}| = r \in \mathbb{N}$.

Theorem 9 *Same assumptions as Theorem 3. We have the following, for $\mathcal{K} = \mathcal{K}_{\mathcal{G}^{(d)}}$ or $\mathcal{K} = \Theta_{\mathcal{G}^{(d)}}$*

$$\mathcal{K}(\boldsymbol{t}) = \sum_{\boldsymbol{r}\in\mathbb{N}^{|\mathcal{N}_0^{(d)}|}} \hat{\mathcal{K}}(\boldsymbol{r})\boldsymbol{P}_{\boldsymbol{r}}(\boldsymbol{t}) \qquad \textit{with} \qquad \hat{\mathcal{K}}(\boldsymbol{r}) \sim d^{-\mathcal{S}(\boldsymbol{r})} \quad \textit{if} \quad \boldsymbol{r}\neq 0 \quad \textit{and} \quad |\boldsymbol{r}| \lesssim 1. \tag{83}$$

Note that Theorem 3 follows from this theorem and the addition theorem.

Proof [Proof of Theorem 3] Assume $\boldsymbol{r} \neq \boldsymbol{0}$. Indeed, setting

$$\boldsymbol{\xi} = (\boldsymbol{\xi}_u)_{u\in\mathcal{N}_0^{(d)}} \in \boldsymbol{\mathcal{X}}, \quad \boldsymbol{\eta} = (\boldsymbol{\eta}_u)_{u\in\mathcal{N}_0^{(d)}} \in \boldsymbol{\mathcal{X}} \quad \textit{and} \quad \boldsymbol{t} = (\boldsymbol{t}_u)_{u\in\mathcal{N}_0^{(d)}} = (\boldsymbol{\xi}_u^T\boldsymbol{\eta}_u/\boldsymbol{d}_u)_{u\in\mathcal{N}_0^{(d)}},$$

we have

$$\boldsymbol{P}_{\boldsymbol{r}}(\boldsymbol{t}) = \prod_{u\in\mathcal{N}_0^{(d)}} P_{\boldsymbol{r}_u}(\boldsymbol{t}_u) = \prod_{u\in\mathcal{N}_0^{(d)}} N(\boldsymbol{d}_u,\boldsymbol{r}_u)^{-1} \sum_{\boldsymbol{l}_u\in N(\boldsymbol{d}_u,\boldsymbol{r}_u)} Y_{\boldsymbol{r}_u,\boldsymbol{l}_u}(\boldsymbol{\xi}_u/\sqrt{\boldsymbol{d}_u}) Y_{\boldsymbol{r}_u,\boldsymbol{l}_u}(\boldsymbol{\eta}_u/\sqrt{\boldsymbol{d}_u}) \tag{84}$$

$$= \boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})^{-1} \sum_{\boldsymbol{l}\in[\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})]} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}). \tag{85}$$

Then Theorem 3 follows by noticing

$$\hat{\mathcal{K}}(\boldsymbol{r})\boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})^{-1} \sim d^{-\mathcal{S}(\boldsymbol{r})} \prod_{u\in\mathcal{N}_0^{(d)}} \boldsymbol{d}_u^{-\boldsymbol{r}_u} \sim d^{-\mathcal{S}(\boldsymbol{r})} d^{-\sum_{u\in\mathcal{N}_0^{(d)}} \alpha_u \boldsymbol{r}_u} = d^{-\mathcal{S}(\boldsymbol{r})-\mathcal{L}(\boldsymbol{r})} = d^{-\mathcal{L}(\boldsymbol{r})}.$$

■

Proof [Proof of Theorem 9] From the orthogonality,

$$\hat{\mathcal{K}}(\boldsymbol{r}) = \langle \mathcal{K}, \boldsymbol{P}_{\boldsymbol{r}}\rangle / \|\boldsymbol{P}_{\boldsymbol{r}}\|^2_{L^2(\boldsymbol{I},\boldsymbol{\omega})} \tag{86}$$

We begin with the denominator. Note that

$$\|\boldsymbol{P}_{\boldsymbol{r}}\|_{L^2(\boldsymbol{I},\boldsymbol{\sigma})}{}^2 = \prod_{u\in\mathcal{N}_0^{(d)}} \|P_{\boldsymbol{r}_u}\|^2_{L^2(I,\omega_{\boldsymbol{d}_u})} = \boldsymbol{N}(\boldsymbol{d};\boldsymbol{r})^{-1} \prod_{u\in\mathcal{N}_0^{(d)}} (|\mathbb{S}_{\boldsymbol{d}_u-1}|/|\mathbb{S}_{\boldsymbol{d}_u-2}|) \tag{87}$$

By applying Lemma 7, integration by parts and continuity of $\mathcal{K}^{(\boldsymbol{r})}$ on the boundary $\partial \boldsymbol{I}$

$$\langle \mathcal{K}, \boldsymbol{P}_{\boldsymbol{r}} \rangle_{L^2(\boldsymbol{I},\boldsymbol{\omega})} = c_{\boldsymbol{r}} \int_{\boldsymbol{I}} \mathcal{K}(\boldsymbol{t}) \left(\frac{d}{d\boldsymbol{t}}\right)^{\boldsymbol{r}} \left(1-\boldsymbol{t}^2\right)^{\boldsymbol{r}+(\boldsymbol{d}-3)/2} d\boldsymbol{t} \tag{88}$$

$$= (-1)^{\boldsymbol{r}} c_{\boldsymbol{r}} \int_{\boldsymbol{I}} \mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{t}) \left(1-\boldsymbol{t}^2\right)^{\boldsymbol{r}+(\boldsymbol{d}-3)/2} d\boldsymbol{t} \tag{89}$$

$$= (-1)^{\boldsymbol{r}} c_{\boldsymbol{r}} \left(\mathcal{M}(\mathcal{K}, \boldsymbol{d}) + \epsilon(\mathcal{K}, \boldsymbol{d})\right) \tag{90}$$

where $\mathcal{K}^{(\boldsymbol{r})}$ is the $\boldsymbol{r}$ derivative of $\mathcal{K}$, the coefficient $c_{\boldsymbol{r}}$ is given by Lemma 7

$$c_{\boldsymbol{r}} = \prod_{u \in \mathcal{N}_0^{(d)}} c_{\boldsymbol{r}_u} = \prod_{u \in \mathcal{N}_0^{(d)}} \frac{(-1)^{\boldsymbol{r}_u}}{2^{\boldsymbol{r}_u} (\boldsymbol{r}_u + (\boldsymbol{d}_u - 3)/2)_{\boldsymbol{r}_u}} \sim \prod_{u \in \mathcal{N}_0^{(d)}} (-1)^{\boldsymbol{r}_u} \boldsymbol{d}_u^{-\boldsymbol{r}_u} = (-1)^{\boldsymbol{r}} \boldsymbol{d}^{-\boldsymbol{r}} \tag{91}$$

(note that only $\lesssim 1$ many $\boldsymbol{r}_u \neq 0$) and the major and error terms are given by

$$\mathcal{M}(\mathcal{K}, \boldsymbol{d}) \equiv \mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{0}) \int_{\boldsymbol{I}} \left(1-\boldsymbol{t}^2\right)^{\boldsymbol{r}+(\boldsymbol{d}-3)/2} d\boldsymbol{t} = \mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{0}) \prod_{u \in \mathcal{N}_0^{(d)}} \frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} \tag{92}$$

$$\epsilon(\mathcal{K}, \boldsymbol{d}) \equiv \int_{\boldsymbol{I}} (\mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{t}) - \mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{0})) \left(1-\boldsymbol{t}^2\right)^{\boldsymbol{r}+(\boldsymbol{d}-3)/2} d\boldsymbol{t} \tag{93}$$

We first show that the error term is small. The mean value theorem gives

$$|\mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{t}) - \mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{0})| \leq \sum_{u \in \mathcal{N}_0^{(d)}} \|\mathcal{K}^{(\boldsymbol{r}+\boldsymbol{e}_u)}\|_{L^\infty(\boldsymbol{I})} |\boldsymbol{t}_u| \tag{94}$$

and the error term $|\epsilon(\mathcal{K}, \boldsymbol{d})|$ is bounded above by

$$\int_{\boldsymbol{I}} \left(1-\boldsymbol{t}^2\right)^{\boldsymbol{r}+(\boldsymbol{d}-3)/2} d\boldsymbol{t} \sum_{u \in \mathcal{N}_0^{(d)}} \|\mathcal{K}^{(\boldsymbol{r}+\boldsymbol{e}_u)}\|_{L^\infty(\boldsymbol{I})} \left(\frac{\int_I |\boldsymbol{t}_u| \left(1-\boldsymbol{t}_u^2\right)^{\boldsymbol{r}_u + (\boldsymbol{d}_u - 3)/2} d\boldsymbol{t}_u}{\int_I \left(1-\boldsymbol{t}_u^2\right)^{\boldsymbol{r}_u + (\boldsymbol{r}_u - 3)/2} d\boldsymbol{t}_u} \right) \tag{95}$$

$$\lesssim \left(\prod_{u \in \mathcal{N}_0^{(d)}} \frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} \right) \left(\sum_{u \in \mathcal{N}_0^{(d)}} \|\mathcal{K}^{(\boldsymbol{r}+\boldsymbol{e}_u)}\|_{L^\infty(\boldsymbol{I})} \boldsymbol{d}_u^{-1} \left(\frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} \right)^{-1} \right) . \tag{96}$$

Since for any $\alpha \in \mathbb{N}$, as $\boldsymbol{d}_u \to \infty$,

$$\frac{|\mathbb{S}_{\alpha + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{\alpha + \boldsymbol{d}_u - 2}|} = \pi^{\frac{1}{2}} \Gamma((\alpha + \boldsymbol{d}_u - 1)/2) / \Gamma((\alpha + \boldsymbol{d}_u)/2) \sim \pi^{\frac{1}{2}} (\boldsymbol{d}_u/2)^{-\frac{1}{2}} \sim (\boldsymbol{d}_u)^{-\frac{1}{2}} , \tag{97}$$

we have

$$|\epsilon(\mathcal{K}, \boldsymbol{d})| \lesssim \sum_{u \in \mathcal{N}_0^{(d)}} \|\mathcal{K}^{(\boldsymbol{r}+\boldsymbol{e}_u)}\|_{L^\infty(\boldsymbol{I})} \boldsymbol{d}_u^{-\frac{1}{2}} \prod_{u \in \mathcal{N}_0^{(d)}} \frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} . \tag{98}$$

We claim that (which will be proved later)

$$\sum_{u \in \mathcal{N}_0^{(d)}} \|\mathcal{K}^{(\boldsymbol{r}+\boldsymbol{e}_u)}\|_{L^\infty(\boldsymbol{I})} \lesssim d^{-\mathcal{S}(\boldsymbol{r})} \tag{99}$$

which implies

$$\langle \mathcal{K}, \boldsymbol{P}_{\boldsymbol{r}} \rangle_{L^2(\boldsymbol{I}, \omega_{d_{\text{in}}}^p)} = c_{\boldsymbol{r}} \left(\mathcal{K}^{(\boldsymbol{r})}(0) + \mathcal{O}\left(d^{-\mathbb{S}(\boldsymbol{r})} (\min_{u \in \mathcal{N}_0^{(d)}} \boldsymbol{d}_u)^{-\frac{1}{2}} \right) \right) \prod_{u \in \mathcal{N}_0^{(d)}} \frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} \tag{100}$$

Plugging back to Eq. (86), we have

$$\hat{\mathcal{K}}(\boldsymbol{r}) = (-1)^{\boldsymbol{r}} c_{\boldsymbol{r}} \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r}) \left(\mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{0}) + \mathcal{O}\left(d^{-\mathbb{S}(\boldsymbol{r})} (\min_{u \in \mathcal{N}_0^{(d)}} \boldsymbol{d}_u)^{-\frac{1}{2}} \right) \right) \left(\prod_{u \in \mathcal{N}_0^{(d)}} \frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} \left(\frac{|\mathbb{S}_{\boldsymbol{d}_u - 1}|}{|\mathbb{S}_{\boldsymbol{d}_u - 2}|} \right)^{-1} \right) \tag{101}$$

Since, for $\boldsymbol{r}$ fixed and as $\boldsymbol{d}_u \to \infty$ for all $u \in \mathcal{N}_0^{(d)}$

$$\frac{c_{\boldsymbol{r}}}{(-1)^{\boldsymbol{r}} \boldsymbol{d}^{-\boldsymbol{r}}} \to 1 \quad \text{and} \quad \frac{\boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r})}{\boldsymbol{d}^{\boldsymbol{r}} / \boldsymbol{r}!} \to 1 \quad \text{and} \quad \left(\prod_{u \in \mathcal{N}_0^{(d)}} \frac{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 1}|}{|\mathbb{S}_{2\boldsymbol{r}_u + \boldsymbol{d}_u - 2}|} \left(\frac{|\mathbb{S}_{\boldsymbol{d}_u - 1}|}{|\mathbb{S}_{\boldsymbol{d}_u - 2}|} \right)^{-1} \right) \to 1 \tag{102}$$

The last limit have used the fact that $|\boldsymbol{r}|$ is bounded (i.e. $\lesssim 1$) and $\boldsymbol{r}_u \neq 0$ for at most $|\boldsymbol{r}|$ many u. Therefore

$$\hat{\mathcal{K}}(\boldsymbol{r}) \sim \boldsymbol{r}!^{-1} \left(\mathcal{K}^{(\boldsymbol{r})}(\boldsymbol{0}) + \mathcal{O}\left(d^{-\mathbb{S}(\boldsymbol{r})} \left(\min_{u \in \mathcal{N}_0^{(d)}} \boldsymbol{d}_u \right)^{-\frac{1}{2}} \right) \right) \tag{103}$$

It remains to verify Eq. (99). By Lemma 6, we only need to show that

$$\sum_{u \in \mathcal{N}_0^{(d)}} d^{-\mathbb{S}(\boldsymbol{r} + \boldsymbol{e}_u)} \lesssim d^{-\mathbb{S}(\boldsymbol{r})} \tag{104}$$

We prove this by induction on the number of hidden layers of $\mathcal{G}^{(d)}$. The base case is obvious. Now suppose the depth of $\mathcal{G}^{(d)}$ is h. Let $\mathcal{C}(\boldsymbol{r})$ be the set of children of $o_{\mathcal{G}}$ that are ancestors of at least one node of $\boldsymbol{n}(\boldsymbol{r})$. We split $\mathcal{N}_0^{(d)}$ into two disjoint sets

$$\mathcal{Q}(\boldsymbol{r}) \equiv \{ u \in \mathcal{N}_0^{(d)} : \exists v \in \mathcal{C}(\boldsymbol{r}) \text{ s.t. } \mathcal{P}(u \to v) \neq \emptyset \} \quad \text{and} \quad \mathcal{N}_0^{(d)} \backslash \mathcal{Q}(\boldsymbol{r}) \,.$$

For $u \notin \mathcal{Q}(\boldsymbol{r})$, we have $\mathbb{S}(\boldsymbol{r} + \boldsymbol{e}_u) = \mathbb{S}(\boldsymbol{r}) + \mathbb{S}(\boldsymbol{e}_u)$ and hence

$$\sum_{u \notin \mathcal{Q}(\boldsymbol{r})} d^{-\mathbb{S}(\boldsymbol{r} + \boldsymbol{e}_u)} = \sum_{u \notin \mathcal{Q}(\boldsymbol{r})} d^{-\mathbb{S}(\boldsymbol{r}) - \mathbb{S}(\boldsymbol{e}_u)} = d^{-\mathbb{S}(\boldsymbol{r})} \sum_{u \notin \mathcal{Q}(\boldsymbol{r})} d^{-\mathbb{S}(\boldsymbol{e}_u)} \lesssim d^{-\mathbb{S}(\boldsymbol{r})} \,. \tag{105}$$

In the last inequality above, we have used

$$\sum_{u \notin \mathcal{Q}(\boldsymbol{r})} d^{-\mathbb{S}(\boldsymbol{e}_u)} \leq \sum_{u \in \mathcal{N}_0^{(d)}} d^{-\mathbb{S}(\boldsymbol{e}_u)} \sim 1 \,. \tag{106}$$

To estimate the remaining, we use induction. Note that $|\mathcal{C}(\boldsymbol{r})|$ is finite (i.e. $\lesssim 1$) and independent of d. Then

$$\sum_{u\in\mathcal{Q}(\boldsymbol{r})} d^{-\mathbb{S}(\boldsymbol{r}+\boldsymbol{e}_u)} \le \sum_{v\in\mathcal{C}(\boldsymbol{r})}\sum_{u\in\mathcal{Q}(\boldsymbol{r})} d^{-\alpha_{o_\mathcal{G}}-\mathbb{S}(n(\boldsymbol{r}+\boldsymbol{e}_u;v))} \tag{107}$$

$$= d^{-\alpha_{o_\mathcal{G}}}\sum_{v\in\mathcal{C}(\boldsymbol{r})}\sum_{u\in\mathcal{Q}(\boldsymbol{r})} d^{-\mathbb{S}(n(\boldsymbol{r}+\boldsymbol{e}_u;v))} \tag{108}$$

$$\lesssim |\mathcal{C}(\boldsymbol{r})| d^{-\alpha_{o_\mathcal{G}}} \max_{v\in\mathcal{C}(\boldsymbol{r})} d^{-\mathbb{S}(n(\boldsymbol{r};v))} \sim d^{-\mathbb{S}(\boldsymbol{r})} \tag{109}$$

We have used induction on the sub-graph with v as the output node. ■

Appendix F. Proof of Theorem 5

Let $\mathcal{G}^{(d)}$ be a DAG associated to the convolutional networks whose filter sizes in the l-th layer is $k_l = [d^{\alpha_l}]$, for $0 \le l \le L+1$, in which we treat the *flatten-dense* readout layer as a convolution with filter size $[d^{\alpha_{L+1}}]$. Note that we have set $\alpha_p = \alpha_0$ and $\alpha_w = \alpha_{L+1}$. We also assume an *activation* layer after the *flatten-dense* layer, which does not essentially alter the topology of the DAG.

We need the following dimension counting lemma.

Lemma 10 *Let $r \in \mathfrak{L}(\mathcal{G}^{(d)})$. Then*

$$\dim\left(\operatorname{span}\left\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}: \quad \mathfrak{L}(\boldsymbol{r}) = r,\ \boldsymbol{l} \ \in \boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})\right\}\right) \sim d^r \tag{110}$$

To prove Theorem 5, we only need to verify the assumptions of Theorem 4 in Mei et al. (2021a). For convenience, we briefly recap the assumptions and results from Mei et al. (2021a) in Sec.I.

It is convenient to group the eigenspaces together according to the learning indices $\mathfrak{L}(\mathcal{G}^{(d)})$. Recall that $\mathfrak{L}(\mathcal{G}^{(d)}) = (r_1 \le r_2 \le r_3 \dots)$. Let

$$E_i = \operatorname{span}\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}} : \mathfrak{L}(\boldsymbol{r}) = r_i\} \tag{111}$$

Then by Theorem 3 and Lemma 10,

$$\dim(E_i) \sim d^{r_i} \quad \text{and} \quad \lambda(g) \sim d^{-r_i} \quad \forall g \in E_i,\ g \ne \mathbf{0}\,, \tag{112}$$

where $\lambda(g)$ denotes the eigenvalue of g. We proceed to verify Assumptions 4 and 5 in Sec. I. They follow directly from Theorem 3, Lemma 10 and the hypercontractivity of spherical harmonics Beckner (1992).

F.1. Verifying Assumption 4

We need the following.

Proposition 11 *For $0 < s \in \mathbb{R}$, let $D_s = \operatorname{span}\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}} : |\mathfrak{L}(\boldsymbol{r})| < s\}$. Then for $\boldsymbol{f} \in D_s$,*

$$\|\boldsymbol{f}\|_q^2 \le (q-1)^{s/\alpha_0}\|\boldsymbol{f}\|_2^2 \tag{113}$$

Proof [Proof of Proposition 11.] The lemma follows from the tensorization of hypercontractivity. Let $f = \sum_{k\geq 0} Y_k \in L^2(\mathbb{S}_n)$ where Y_k is a degree k spherical harmonics in $\mathbb{S}_n$. Define the Poisson semi-group operator

$$P_\epsilon f(x) = \sum_{k\geq 0} \epsilon^k Y_k(x) \tag{114}$$

Then we have the hypercontractivity inequality (Beckner, 1992), for $1 \leq p \leq q$ and $\epsilon \leq \sqrt{\frac{p-1}{q-1}}$

$$\|P_\epsilon f\|_{L^q(\mathbb{S}_n)} \leq \|f\|_{L^p(\mathbb{S}_n)} \tag{115}$$

One can then tensorize (Beckner, 1975) it to obtain the same bound in the tensor space.

Lemma 12 (Corollary 11 Montanaro (2012)) *Let $\boldsymbol{f} : (\mathbb{S}_n)^k \to \mathbb{R}$. If $1 \leq p \leq q$ and $\epsilon \leq \sqrt{\frac{p-1}{q-1}}$, then*

$$\|P_\epsilon^{\otimes k} \boldsymbol{f}\|_{L^q((\mathbb{S}_n)^k)} \leq \|\boldsymbol{f}\|_{L^p((\mathbb{S}_n)^k)} . \tag{116}$$

Let $\boldsymbol{f} = \sum_{\boldsymbol{r},\boldsymbol{l}} a_{\boldsymbol{r},\boldsymbol{l}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}} \in D_s$. Choosing $\epsilon = \sqrt{\frac{1}{q-1}}$ and $p = 2$ in the above lemma, we have

$$\begin{aligned}
\|\boldsymbol{f}\|_q^2 &= \|\sum_{\boldsymbol{r},\boldsymbol{l}} a_{\boldsymbol{r},\boldsymbol{l}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\|_q^2 && (117)\\
&= \|P_\epsilon^{\otimes |\mathcal{N}_0^{(d)}|} \sum_{\boldsymbol{r},\boldsymbol{l}} a_{\boldsymbol{r},\boldsymbol{l}} \epsilon^{-\boldsymbol{r}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\|_q^2 && (118)\\
&\leq \|\sum_{\boldsymbol{r},\boldsymbol{l}} a_{\boldsymbol{r},\boldsymbol{l}} \epsilon^{-\boldsymbol{r}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\|_2^2 && (119)\\
&= \sum_{\boldsymbol{r},\boldsymbol{l}} a_{\boldsymbol{r},\boldsymbol{l}}^2 \epsilon^{-2\boldsymbol{r}} \|\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\|_2^2 && (120)\\
&\leq \epsilon^{-2\max|\boldsymbol{r}|} \sum_{\boldsymbol{r},\boldsymbol{l}} a_{\boldsymbol{r},\boldsymbol{l}}^2 \|\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}\|_2^2 && (121)\\
&= (q-1)^{\max|\boldsymbol{r}|} \|\boldsymbol{f}\|_2^2 \leq (q-1)^{s/\alpha_0} \|\boldsymbol{f}\|_2^2 && (122)
\end{aligned}$$

■

Since $r \notin \mathfrak{L}(\mathcal{G}^{(d)})$, there is a j such that $r_j < r < r_{j+1}$. Let $n(d) = d^r$ and

$$m(d) = \dim\left(\text{span}\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}} : \mathfrak{L}(\boldsymbol{r}) \leq r_j\}\right) = \dim(\text{span} \bigcup_{i\leq j} E_i) \tag{123}$$

Clearly, $m(d) \sim d^{r_j}$. We list all eigenvalues of $\mathcal{K}$ in non-ascending order as $\{\lambda_{d,i}\}$. In particular, we have

$$\lambda_{d,m(d)} \sim d^{-r_j} > d^{-r} > d^{-r_{j+1}} \sim \lambda_{d,m(d)+1} . \tag{124}$$

Assumption 4 (a). We choose $u(d)$ to be

$$u(d) = \dim\left(\text{span} \bigcup_{i:r_i\leq 2r+100} E_i\right) . \tag{125}$$

Assumption 4 (a) follows from Proposition 11.

Assumption 4 (b). Let $s = \inf\{\bar{r} \in \mathcal{Y}(\mathcal{G}^{(d)}) : \bar{r} > 2r + 100\}$. For $l > 1$, we have

$$\sum_{j=u(d)+1} \lambda_{d,j}^l \sim \sum_{r_i : r_i \geq s} (d^{-r_i})^l \dim(E_i) \sim d^{-s(l-1)} \tag{126}$$

which also holds for $l = 1$ since

$$\sum_{j=u(d)+1} \lambda_{d,j} \sim 1 \tag{127}$$

Thus

$$\frac{(\sum_{j=u(d)+1} \lambda_{d,j}^l)^2}{\sum_{j=u(d)+1} \lambda_{d,j}^{2l}} \sim \frac{d^{-2s(l-1)}}{d^{-s(2l-1)}} = d^s > d^{2r+100} > n(d)^{2+\delta} \sim d^{(2+\delta)r}. \tag{128}$$

as long as $\delta < 100/r$.

Assumption 4 (c). Denote

$$\mathcal{K}_{d,>m(d)}(\boldsymbol{\xi}, \boldsymbol{\eta}) = \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}) \tag{129}$$

We have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\eta}} \mathcal{K}_{d,>m(d)}(\boldsymbol{\xi}, \boldsymbol{\eta})^2 =& \mathbb{E}_{\boldsymbol{\eta}} \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta})|^2 && (130)\\
=& \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})|^2 && (131)\\
=& \sum_{\boldsymbol{r} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 \sum_{\boldsymbol{l}} |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})|^2 && (132)\\
=& \sum_{\boldsymbol{r} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r}) \boldsymbol{P}_{\boldsymbol{r}}(\boldsymbol{1}) = \sum_{\boldsymbol{r} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r}) \,. && (133)
\end{aligned}$$

Similarly,

$$\mathbb{E}_{\boldsymbol{\xi},\boldsymbol{\eta}} \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta})|^2 = \sum_{\boldsymbol{r} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r}) \,. \tag{134}$$

Thus

$$\mathbb{E}_{\boldsymbol{\eta}} \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta})|^2 - \mathbb{E}_{\boldsymbol{\xi},\boldsymbol{\eta}} \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r})^2 |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta})|^2 = 0 \,. \tag{135}$$

For the diagonal terms,

$$\mathcal{K}_{d,>m(d)}(\boldsymbol{\xi}, \boldsymbol{\xi}) = \sum_{\boldsymbol{r},\boldsymbol{l} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r}) |\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})|^2 = \sum_{\boldsymbol{r} : \mathcal{Y}(\boldsymbol{r}) > r} \lambda_{\mathcal{K}}(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r}) = \mathbb{E}_{\boldsymbol{\xi}} \mathcal{K}_{d,>m(d)}(\boldsymbol{\xi}, \boldsymbol{\xi}) \tag{136}$$

which is deterministic.

F.2. Verifying Assumption 5.

Recall that $n(d) \sim d^r$ and $r_j < r < r_{j+1}$. **Assumption 5(a)** follows from Eq. (112). Indeed, for $l > 1$

$$\lambda_{d,m(d)+1}^{-l} \sum_{k=m(d)+1} \lambda_{d,k}^{l} \sim (d^{-r_{j+1}})^{-l} \sum_{i:r_i \geq r_{j+1}} \dim(E_i) d^{-lr_i} \tag{137}$$

$$\sim d^{lr_{j+1}} \sum_{i:r_i \geq r_{j+1}} d^{r_i} d^{-lr_i} \tag{138}$$

$$= d^{r_{j+1}} > n(d)^{1+\delta} \tag{139}$$

for some $\delta > 0$ since $r < r_{j+1}$. Similarly, for $l = 1$, since

$$\sum_{k=m(d)+1} \lambda_{d,k} \sim 1 \tag{140}$$

we have

$$(d^{-r_{j+1}})^{-1} \sum_{k=m(d)+1} \lambda_{d,k} \sim d^{r_{j+1}} > n(d)^{1+\delta} . \tag{141}$$

Assumption 5(b) follows from $r > r_j$.

Assumption 5(c). Note that

$$\lambda_{d,m(d)}^{-1} \sum_{k=m(d)+1} \lambda_{d,k} \sim \lambda_{d,m(d)}^{-1} \sim d^{r_j} < n(d)^{(1-\delta)} \sim d^{r(1-\delta)} \tag{142}$$

for some $\delta > 0$ since $r_j < r$.

Appendix G. Proof of Lemma 10

Proof The lemma can be proved by induction. **Base case: $L = 0$.** The network is a S-CNN. WLOG, assume $\alpha_0 \neq 0$ and $\alpha_1 \neq 0$. For $r \in \mathfrak{L}(\mathcal{G}^{(d)})$, we know that r can be written as a combination of α_0 and α_1, i.e. $r = k_0\alpha_0 + k_1\alpha_1$ for some $k_0, k_1 \geq 0$. We say a tuple (k_0, k_1) is r-feasible if in addition, there exists $\boldsymbol{r}$ with $\mathfrak{S}(\boldsymbol{r}) = k_1\alpha_1$ and $\mathfrak{F}(\boldsymbol{r}) = k_0\alpha_0$. Consider the set of all r-feasible tuple

$$F(r) \equiv \{(k_0, k_1) : r\text{-feasible}\} . \tag{143}$$

Clearly, $F(r)$ is finite. It suffices to prove that for each r-feasible tuple (k_0, k_1),

$$\dim\left(\operatorname{span}\left\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}} : \mathfrak{F}(\boldsymbol{r}) = k_0\alpha_0 \quad \mathfrak{S}(\boldsymbol{r}) = k_1\alpha_1,\ \boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r})\right\}\right) \sim d^r \tag{144}$$

Note that there are $\sim \ (d^{\alpha_1})^{k_1}$ many ways to choose k_1 nodes in the penultimate layer. Then the dimension of the above set is about

$$(d^{\alpha_1})^{k_1} \sum_{\substack{(k_{0,1},\ldots,k_{0,k_1}) \\ k_{0,1}+\cdots+k_{0,k_1}=k_0}} \prod_{j=1}^{k_1} N(d^{\alpha_0}, k_{0,j}) \sim (d^{\alpha_1})^{k_1} \sum_{\substack{(k_{0,1},\ldots,k_{0,k_1}) \\ k_{0,1}+\cdots+k_{0,k_1}=k_0}} \prod_{j=1}^{k_1} (d^{\alpha_0})^{k_{0,j}} \tag{145}$$

$$\sim d^{k_0\alpha_0+k_1\alpha_1} = d^r , \tag{146}$$

since the cardinality of the set

$$\{(k_{0,1},\dots,k_{0,k_1}) : k_{0,1}+\cdots+k_{0,k_1}=k_0, \quad k_{0,j}\geq 1 \quad k_{0,j}\in\mathbb{N}\}$$

is finite.

Induction step: $L \geq 1$**.** For $\boldsymbol{r}$ with $\mathscr{L}(\boldsymbol{r}) = r$, let k be the number of children of a MST of $n(\boldsymbol{r}; o_{\mathcal{G}})$. Clearly, $k \in [1, [r/\alpha_{L+1}]]$. Then we can classify $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ into at most $[r/\alpha_{L+1}]$ bins: $\{\Omega_k\}_{k=1,\dots,[r/\alpha_{L+1}]}$ depending on the number of children of $o_{\mathcal{G}}$ in a MST. Let Ω_k be non-empty. We only need to prove the number of $\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}$ in Ω_k is d^r. Note that there are $\sim (d^{\alpha_{L+1}})^k$ many ways to choose k children from $o_{\mathcal{G}}$. Let $\{u_j\}_{j=1,\dots,k}$ be one fixed choice and $\{\mathcal{G}_j\}$ be the subgraphs with $\{u_j\}$ as the output nodes. Next, we partition $(r - k\alpha_{L+1})$ into k components,

$$r - k\alpha_{L+1} = r_1 + \cdots + r_k$$

so that each r_j is a combination of $\{\alpha_j\}_{0\leq j\leq L}$. The cardinality of such partition is also finite. We fix one of such partition $(r_1,\dots,r_k)$ so that each r_j is a learning index of $\mathcal{G}_j$. We can apply induction to each $(\mathcal{G}_j, r_j)$ to conclude that the cardinality of $\overline{\boldsymbol{Y}}_{\boldsymbol{r}_j,\boldsymbol{l}_j}$ with $\mathscr{L}_{\mathcal{G}_j}(\boldsymbol{r}_j) = r_j$ is $\sim d^{r_j}$, where $\mathscr{L}_{\mathcal{G}_j}(\boldsymbol{r}_j)$ is the learning index of $\boldsymbol{r}_j$ of $\mathcal{G}_j$. Therefore, we have

$$\dim\left(\text{span}\left\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}} : \quad \mathscr{L}(\boldsymbol{r}) = r,\ \boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d},\boldsymbol{r})\right\}\right) \sim (d^{\alpha_{L+1}})^k \prod_{j\in[k]} d^{r_j} = d^r\,. \tag{147}$$

■

Appendix H. CNN-GAP: CNNs with Global Average Pooling

Consider convolutional networks whose readout layer is a global average pooling (GAP) and a flattening layer (namely, without pooling), resp.

$$\textbf{CNN + GAP:} \qquad [\textit{Conv}(p)\textit{-Act}] \to [\textit{Conv}(k)\textit{-Act}]^{\otimes L} \to [\textit{GAP}] \to [\textit{Dense}] \tag{148}$$

$$\textbf{CNN:} \qquad [\textit{Conv}(p)\textit{-Act}] \to [\textit{Conv}(k)\textit{-Act}]^{\otimes L} \to [\textit{Flatten}] \to [\textit{Dense}] \tag{149}$$

Concretely, the input space is $\boldsymbol{\mathcal{X}} = (\overline{\mathbb{S}}_{p-1})^{k^L\times w} \subseteq \mathbb{R}^{p\times 1\times k^L\times w}$, where p is the patch size of the input convolutional layer, k is the filter size in *hidden* layers, L is the number of *hidden* convolution layers and w is the spatial dimension of the penultimate layer. The total dimension of the input is $d = p\cdot k^L\cdot w$, and the number of input nodes is $|\mathcal{N}_0| = k^L\cdot w$. Since the stride is equal to the filter size for all convolutional layers, the *spatial* dimension is reduced by a factor of p in the first layer, a factor of k by each hidden layer. The penultimate layer (before pooling/flattening) has spatial dimension w and is reduced to 1 by the *GAP-dense* layer or the *Flatten-dense* layer. The DAGs associated to these two architectures are identical which is denoted by $\mathcal{G}$. However, the kernel/neural network computations are slightly different. If the penultimate layer has n many channels and $f_{pen} : \boldsymbol{\mathcal{X}} \to \mathbb{R}^{n\times w}$ is the mapping from the input layer to the penultimate layer, then the outputs of the *CNN-GAP* and *CNN-Flatten* are

$$f_{\text{CNN-GAP}}(\boldsymbol{x}) = n^{-\frac{1}{2}} \sum_{j\in[n]} \omega_j \fint_{i\in[w]} f_{pen}(\boldsymbol{x})_{j,i} \tag{150}$$

$$f_{\text{CNN-Flatten}}(\boldsymbol{x}) = (nw)^{-\frac{1}{2}} \sum_{j\in[n],i\in[w]} \omega_{ji} f_{pen}(\boldsymbol{x})_{j,i}\,, \tag{151}$$

resp., where w_j and w_{ji} are parameters of the last layer and are usually initialized with standard iid Guassian $w_j, w_{ji} \sim \mathcal{N}(0,1)$. Let $\mathcal{N}_{-1} \subseteq \mathcal{N}$ be the nodes in the penultimate layer, then $|\mathcal{N}_{-1}| = w$. Let $\boldsymbol{\xi} = (\boldsymbol{\xi}_v)_{v\in\mathcal{N}_{-1}} \in \boldsymbol{\mathcal{X}}$, where $\boldsymbol{\xi}_v \in (\overline{\mathbb{S}}_{p-1})^{k^L}$. Thus, each $\boldsymbol{\xi}_v$ contains k^L many input nodes $\{\boldsymbol{\xi}_{u,i}\}_{i\in[k^L]}$. Define

$$\boldsymbol{t}_{uv} = (\langle \boldsymbol{\xi}_{u,i}, \boldsymbol{\eta}_{v,i}\rangle/p)_{i\in[k^L]} \in [-1,1]^{k^L}. \tag{152}$$

Recall that in the case without pooling

$$\mathcal{K}_u(\boldsymbol{t}) = \phi^*(\fint_{uv\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t})), \quad \mathcal{K}_{\mathcal{G}} = \fint_{o_{\mathcal{G}}v\in\mathcal{E}} \mathcal{K}_v(\boldsymbol{t}) = \fint_{v\in\mathcal{N}_{-1}} \mathcal{K}_v(\boldsymbol{t}) \tag{153}$$

where $\boldsymbol{t} \in [-1,1]^{k^L\times w}$, which is obtained by $\boldsymbol{t} = (\langle \boldsymbol{\xi}_{u,i}, \boldsymbol{\eta}_{u,i}\rangle/p)_{u\in\mathcal{N}_{-1},i\in[k^L]}$. Indeed, for each $v \in \mathcal{N}_{-1}$, $\mathcal{K}_v$ depends only on the *diagonal* terms $\boldsymbol{t}_{vv} = (\langle \boldsymbol{\xi}_{v,i}, \boldsymbol{\eta}_{v,i}\rangle/p)_{i\in[k^L]} \in [-1,1]^{k^L}$. We can find a function

$$\mathcal{K}_{pen} : [-1,1]^{k^L} \to [-1,1] \quad \text{s.t.} \quad \mathcal{K}_v(\boldsymbol{t}) = \mathcal{K}_{pen}(\boldsymbol{t}_{vv}) \quad \forall v \in \mathcal{N}_{-1} \tag{154}$$

Therefore, without pooling the NNGP kernel is

$$\mathcal{K}_{\text{CNN-Flatten}}(\boldsymbol{t}) = \fint_{v\in\mathcal{N}_{-1}} \mathcal{K}_{pen}(\boldsymbol{t}_{vv}) = \frac{1}{w}\sum_{v\in\mathcal{N}_{-1}} \mathcal{K}_{pen}(\boldsymbol{t}_{vv}) \tag{155}$$

Note that the kernel does not depend on any *off-diagonal* terms $\boldsymbol{t}_{uv}$ with $u \neq v$ because there isn't weight-sharing in the last layer. Let $\boldsymbol{d}_{pen} = (p,p,\ldots,p) \in \mathbb{N}^{k^L}$. Then $\|\boldsymbol{d}_{pen}\|_1 = pk^L$ is the effective dimension of the input to any node $u \in \mathcal{N}_{-1}$. Assume $k = d^{\alpha_k}$, $p = d^{\alpha_p}$ and $w = d^{\alpha_w}$ and $\alpha_k, \alpha_p, \alpha_w > 0$. Applying Theorem 3 to $\mathcal{K}_{pen}$, we have

$$\mathcal{K}_{\text{CNN-Flatten}}(\boldsymbol{t}) = \sum_{\boldsymbol{r}\in\mathbb{N}^{k^L}} \frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) \sum_{v\in\mathcal{N}_{-1}} \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_v), \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}_v) \tag{156}$$

Clearly, the eigenfunctions are $\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_v)\}_{\boldsymbol{r},\boldsymbol{l},v}$ and the corresponding eigenvalues are $\{\frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r})\}_{\boldsymbol{r},\boldsymbol{l},v}$. Note that

$$\frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) = \lambda_{\mathcal{K}_{\mathcal{G}}}(\boldsymbol{r}) \sim d^{-\mathfrak{L}(\boldsymbol{r})} \tag{157}$$

Here and in what follows, we also treat $\boldsymbol{r} \in \mathbb{N}^{k^L}$ as an element in $\mathbb{N}^{wk^L}$.
When the readout layer is a GAP, the weights of the penultimate layer are shared across different spatial locations, namely, all nodes in $\mathcal{N}_{-1}$ use the same weight. As such, the kernel corresponding to the CNN-GAP depends on both the diagonal and off-diagonal terms $\boldsymbol{t} = (\boldsymbol{t}_{uv})_{u,v\in\mathcal{N}_{-1}} \in [-1,1]^{k^L\times k^L}$, which can be written as (Novak et al., 2019c)

$$\mathcal{K}_{\text{CNN-GAP}}(\boldsymbol{t}) = \fint_{u,v\in\mathcal{N}_{-1}} \mathcal{K}_{pen}(\boldsymbol{t}_{uv}) = \frac{1}{w^2}\sum_{u,v\in\mathcal{N}_{-1}} \mathcal{K}_{pen}(\boldsymbol{t}_{uv}) \tag{158}$$

Applying Theorem 3 to $\mathcal{K}_{pen}$, we have

$$\mathcal{K}_{\text{CNN-GAP}}(\boldsymbol{t}) = \frac{1}{w^2} \sum_{\boldsymbol{r}\in\mathbb{N}^{kL}} \lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) \sum_{u,v\in\mathcal{N}_{-1}} \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_u), \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}_v) \tag{159}$$

$$= \sum_{\boldsymbol{r}\in\mathbb{N}^{kL}} \frac{1}{w} \lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \left(w^{-\frac{1}{2}} \int_{u\in\mathcal{N}_{-1}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_u) \right) \left(w^{-\frac{1}{2}} \int_{u\in\mathcal{N}_{-1}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}_u) \right) \tag{160}$$

$$= \sum_{\boldsymbol{r}\in\mathbb{N}^{kL}} \frac{1}{w} \lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}) \tag{161}$$

where

$$\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \equiv w^{-\frac{1}{2}} \int_{u\in\mathcal{N}_{-1}} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_u) \quad \text{, for} \quad \boldsymbol{r}\in\mathbb{N}^{kL} \text{ and } \quad \boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r}) \tag{162}$$

That is the eigenfunctions and eigenvalues of $\mathcal{K}_{\text{CNN-GAP}}$ are $\{\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})\}_{\boldsymbol{r},\boldsymbol{l}}$ and $\{\frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r})\}_{\boldsymbol{r},\boldsymbol{l}}$ resp.

In sum, the eigenvalues of $\mathcal{K}_{\text{CNN-Flatten}}$ and $\mathcal{K}_{\text{CNN-GAP}}$ are the same (up to the multiplicity factor w). Each eigenspace of $\mathcal{K}_{\text{CNN-GAP}}$ is given by symmetric polynomials of the form Eq. (162). We can see that the GAP reduces the dimension of each eigenspace by a factor of w. Same arguments can also be applied to the NTKs (Novak et al., 2019b; Yang, 2019)

$$\Theta_{\text{CNN-Flatten}}(\boldsymbol{t}) = \fint_{v\in\mathcal{N}_{-1}} (\mathcal{K}_{pen}(\boldsymbol{t}_{vv}) + \Theta_{pen}(\boldsymbol{t}_{vv})) \tag{163}$$

$$= \sum_{\boldsymbol{r}\in\mathbb{N}^{kL}} \left(\frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) + \frac{1}{w}\lambda_{\Theta_{pen}}(\boldsymbol{r}) \right) \sum_{v\in\mathcal{N}_{-1}} \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_v), \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}_v) \tag{164}$$

$$\Theta_{\text{CNN-GAP}}(\boldsymbol{t}) = \fint_{u,v\in\mathcal{N}_{-1}} (\mathcal{K}_{pen}(\boldsymbol{t}_{uv}) + \Theta_{pen}(\boldsymbol{t}_{uv})) \tag{165}$$

$$= \sum_{\boldsymbol{r}\in\mathbb{N}^{kL}} \left(\frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) + \frac{1}{w}\lambda_{\Theta_{pen}}(\boldsymbol{r}) \right) \sum_{v\in\mathcal{N}_{-1}} \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}), \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}) \tag{166}$$

where Θ_{pen} is the NTK of the penultimate layer which is the same for all nodes in $\mathcal{N}_{-1}$. Since $\frac{1}{w}\lambda_{\Theta_{pen}} \sim d^{-\mathfrak{L}(\boldsymbol{r})}$, we have

$$\frac{1}{w}\lambda_{\mathcal{K}_{pen}}(\boldsymbol{r}) + \frac{1}{w}\lambda_{\Theta_{pen}}(\boldsymbol{r}) \sim d^{-\mathfrak{L}(\boldsymbol{r})} \tag{167}$$

H.1. Generalization bound of CNN-GAP

We show that GAP improves the data efficiency of D-CNNs by a factor of $w \sim d^{\alpha_w}$ under a stronger assumptions on activations ϕ. However, in the infinite-data-regime, there is no improvement from GAP.

Assumption Poly-ϕ: There is a sufficiently large $J \in \mathbb{N}$ such that for all hiddens nodes u

$$\phi_u^{*(j)}(0) \neq 0 \quad \text{for} \quad 1 \leq j \leq J \quad \text{and} \quad \phi_u^{*(j)}(0) = 0 \quad \text{otherwise} \tag{168}$$

This assumption implies that there are $0 < J_1 < J_2 \in \mathbb{R}$ such that for all $\mathbf{0} \neq \boldsymbol{r}$ with $|\boldsymbol{r}| < J_1$

$$\frac{d^{\boldsymbol{r}}}{d\boldsymbol{t}}\mathcal{K}_{pen}(\mathbf{0}) \neq 0 \quad \text{and} \quad \frac{d^{\boldsymbol{r}}}{d\boldsymbol{t}}\Theta_{pen}(\mathbf{0}) \neq 0 \tag{169}$$

and for all $\boldsymbol{r}$ with $|\boldsymbol{r}| > J_2$

$$\frac{d^{\boldsymbol{r}}}{d\boldsymbol{t}}\mathcal{K}_{pen} \equiv \mathbf{0} \quad \text{and} \quad \frac{d^{\boldsymbol{r}}}{d\boldsymbol{t}}\Theta_{pen} \equiv \mathbf{0} \tag{170}$$

Moreover, $J_1 \to \infty$ as $J \to \infty$.

Let $L^p_{\text{Sym}}(\boldsymbol{\mathcal{X}}) \leq L^p(\boldsymbol{\mathcal{X}})$ be the close subspace spanned by symmetric eigenfunctions Eq. (162). Let $\mathcal{K}_{\text{Sym}} = \mathcal{K}_{\text{CNN-GAP}}$ or $\Theta_{\text{CNN-GAP}}$.

For $X \subseteq \boldsymbol{\mathcal{X}}$ and $r \notin \mathcal{L}(\mathcal{G}^{(d)})$, define the regressor and the projection operator to be

$$\begin{aligned}
\boldsymbol{R}^{\text{Sym}}_X(f)(x) &= \mathcal{K}_{\text{Sym}}(x, X)\mathcal{K}_{\text{Sym}}(X, X)^{-1}f(X) \\
\boldsymbol{P}^{\text{Sym}}_{>r}(f) &= \sum_{\boldsymbol{r}:\mathcal{L}(\boldsymbol{r})>r} \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \langle f, \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}\rangle_{L^2(\boldsymbol{\mathcal{X}})}\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}} \\
\boldsymbol{\mathcal{F}}^{\text{Sym}}_t(f) &= (\mathbf{Id} - e^{-t\mathcal{K}_{\text{Sym}}})f
\end{aligned}$$

Theorem 13 *Let $\boldsymbol{\mathcal{G}} = \{\mathcal{G}^{(d)}\}_d$, where each $\mathcal{G}^{(d)}$ is a DAG associated to the D-CNN in Eq. (148) with $\alpha_k, \alpha_p, \alpha_w > 0$. Let $r \notin \mathcal{L}(\mathcal{G}^{(d)})$ be fixed and the activations satisfy* **Assumption Poly-ϕ** *for $J = J(r)$ sufficiently large. Let $f \in L^2_{Sym}(\boldsymbol{\mathcal{X}})$ with $\mathbb{E}_{\boldsymbol{\sigma}} f = 0$. Then for $\epsilon > 0$,*

$$\left| \left\| \boldsymbol{R}^{Sym}_X(f) - f \right\|^2_{L^2_{Sym}(\boldsymbol{\mathcal{X}})} - \left\| \boldsymbol{P}^{Sym}_{>r}(f) \right\|^2_{L^2_{Sym}(\boldsymbol{\mathcal{X}})} \right| = c_{d,\epsilon}\|f\|^2_{L^{2+\epsilon}_{Sym}(\boldsymbol{\mathcal{X}})}, \tag{171}$$

where $c_{d,\epsilon} \to 0$ in probability as $d \to \infty$ over $X \sim \boldsymbol{\sigma}^{[d^{r-\alpha_w}]}$.

Theorem 14 *Assume* **Assumption-$\boldsymbol{\mathcal{G}}$** *and* **Assumption-ϕ**. *Let $r \notin \mathfrak{L}(\mathcal{G}^{(d)})$ and $t \sim d^r$. Then for $0 < \epsilon < \inf\{|r - \bar{r}| : \bar{r} \in \mathfrak{L}(\mathcal{G}^{(d)})\}$ and $f \in L^2_{Sym}(\boldsymbol{\mathcal{X}})$ with $\mathbb{E}_{\boldsymbol{\sigma}} f = 0$, we have*

$$\|\boldsymbol{\mathcal{F}}^{Sym}_t(\boldsymbol{P}^{Sym}_{<r} f) - \boldsymbol{P}^{Sym}_{<r} f\|_2^2 \lesssim e^{-d^\epsilon}\|\boldsymbol{P}^{Sym}_{<r} f\|_2^2 \quad \text{and} \quad \|\boldsymbol{\mathcal{F}}^{Sym}_t(\boldsymbol{P}^{Sym}_{>r} f) - \boldsymbol{P}^{Sym}_{>r} f\|_2^2 \gtrsim e^{-d^{-\epsilon}}\|\boldsymbol{P}^{Sym}_{>r} f\|_2^2 . \tag{172}$$

Theorem 14 follows directly from the eigendecomposition of the kernels $\mathcal{K}_{\text{Sym}}$. We only need to prove of Theorem 13.

Proof [Proof of Theorem 13] We need the following dimension counting lemma, which follows directly from Lemma 10.

Lemma 15 *Let $r \in \mathfrak{L}(\mathcal{G}^{(d)})$. Then*

$$\dim\left(\text{span}\left\{\overline{\boldsymbol{Y}}^{Sym}_{\boldsymbol{r},\boldsymbol{l}} : \quad \mathcal{L}(\boldsymbol{r}) = r,\ \boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{r})\right\}\right) \sim d^{r-\alpha_w} \tag{173}$$

Recall that $\mathfrak{L}(\mathcal{G}^{(d)}) = \{r_j\}$ in non-descending order. Similarly, let

$$E^{\text{Sym}}_i = \text{span}\{\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}} : \quad \mathfrak{L}(\boldsymbol{r}) = r_i\} \tag{174}$$

From the above lemma, we have

$$\dim(E_i^{\text{Sym}}) \sim d^{r_i-\alpha_w} \tag{175}$$

Since $r \notin \mathfrak{L}(\mathcal{G}^{(d)})$, there is a j such that $r_j < r < r_{j+1}$. Let $n(d) = d^{r-\alpha_w}$ and

$$m(d) = \dim\left(\text{span}\{\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}^{\text{Sym}} : \mathfrak{L}(\boldsymbol{r}) \le r_j\}\right) = \dim(\text{span}\bigcup_{i\le j} E_i^{\text{Sym}}) \tag{176}$$

Clearly, $m(d) \sim d^{r_j-\alpha_w}$. We list all eigenvalues of $\mathcal{K}_{\text{Sym}}$ in non-ascending order as $\{\lambda_{d,i}\}$. In particular, we have

$$\lambda_{d,m(d)} \sim d^{-r_j} > d^{-r} > d^{-r_{j+1}} \sim \lambda_{d,m(d)+1}\,. \tag{177}$$

We proceed to verify Assumptions 4 and 5 in Sec. I.

Assumptions 4 (a) We choose $u(d)$ to be

$$u(d) = \dim\left(\text{span}\bigcup_{i:r_i\le 2r+100} E_i^{\text{Sym}}\right)\,. \tag{178}$$

Let $s = \inf\{\bar{r} \in \mathfrak{L}(\mathcal{G}^{(d)}) : \bar{r} > 2r+100\}$. Assumption 4 (a) follows from Proposition 11.

Assumptions 4 (b) For $l > 1$, we have

$$\sum_{j=u(d)+1} \lambda_{d,j}^l \sim \sum_{r_i:r_i\ge s} (d^{-r_i})^l \dim(E_i^{\text{Sym}}) \sim d^{-s(l-1)-\alpha_w} \tag{179}$$

which also holds for $l = 1$ since

$$d^{\alpha_w} \sum_{j=u(d)+1} \lambda_{d,j} \sim 1 \tag{180}$$

Thus

$$\frac{(\sum_{j=u(d)+1} \lambda_{d,j}^l)^2}{\sum_{j=u(d)+1} \lambda_{d,j}^{2l}} \sim \frac{d^{-2s(l-1)-2\alpha_w}}{d^{-s(2l-1)-\alpha_w}} = d^{s-\alpha_w} > d^{2r+100-\alpha_w} > n(d)^{2+\delta} \sim d^{(2+\delta)(r-\alpha_w)}. \tag{181}$$

Assumption 4 (c) This requires some work and is verified in Sec. H.2.

Assumption 5 (a) Since $m(d) = \dim(\text{span}\bigcup_{i\le j} E_i)$ and $r_{j+1} > r > r_j$, we have

$$\frac{1}{\lambda_{d,m(d)+1}} \sum_{j\ge m(d)+1} \lambda_{d,j}^l \sim \frac{1}{(d^{-r_{j+1}})^l} \sum_{i>j} (d^{-r_i})^l \dim(E_i^{\text{Sym}}) \tag{182}$$

$$\sim \dim(E_{j+1}^{\text{Sym}}) = d^{r_{j+1}-\alpha_w} \tag{183}$$

$$> n(d)^{1+\delta} = d^{(r-\alpha_w)(1+\delta)} \tag{184}$$

as long as $\delta < (r_{j+1}-\alpha_w)/(r-\alpha_w) - 1$.

Assumption 5 (b) This is obvious since $m(d) \sim d^{r_j - \alpha_w}$, $n(d) \sim d^{r-\alpha_w}$ and $r > r_j$.
Assumption 5 (c) This follows from $r_j < r$. Indeed,

$$\frac{1}{\lambda_{d,m(d)}} \sum_{j \geq m(d)+1} \lambda_{d,j} \sim \frac{1}{(d^{-r_j})} \sum_{i>j} (d^{-r_i}) \dim(E_i^{\text{Sym}}) \sim d^{r_j - \alpha_w} \tag{185}$$

$$\leq n(d)^{1-\delta} = d^{(r-\alpha_w)(1-\delta)} \tag{186}$$

as long as $0 < \delta < 1 - (r_j - \alpha_w)/(r - \alpha_w)$. ■

H.2. Verification of Assumptions 4(c).

We begin with proving Eq. (230). Let X_i define the random variable

$$X_i \equiv \mathbb{E}_{\boldsymbol{\xi} \sim \boldsymbol{\sigma}_d} \mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi}_i, \boldsymbol{\xi})^2 \quad \text{and} \quad \Delta_i \equiv X_i - \mathbb{E}X_i \tag{187}$$

where

$$\mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi}, \boldsymbol{\eta}) = \sum_{\boldsymbol{r},\boldsymbol{l}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\eta}) \tag{188}$$

and $\lambda_{\mathcal{K}_{\text{Sym}}}$ is the eigenvalue of $\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}$. We need to show that

$$\frac{\sup_{i \in [n(d)]} |\Delta_i|}{\mathbb{E}X_i} \xrightarrow[d\to\infty]{\text{in prob.}} 0. \tag{189}$$

By Markov's inequality, it suffices to show that

$$(\mathbb{E}X_i)^{-1} \mathbb{E} \sup_{i \in [n(d)]} |\Delta_i| \xrightarrow[d\to\infty]{} 0 \tag{190}$$

By orthogonality and treating $\boldsymbol{r} \in \mathbb{N}^{k^L}$ as an element of $\mathbb{N}^{wk^L}$, we have

$$\mathbb{E}X_i = \mathbb{E}_{\boldsymbol{\xi},\bar{\boldsymbol{\xi}} \sim \boldsymbol{\sigma}_d} \mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi}, \bar{\boldsymbol{\xi}})^2 \tag{191}$$

$$= \mathbb{E}_{\boldsymbol{\xi},\bar{\boldsymbol{\xi}} \sim \boldsymbol{\sigma}_d} \left| \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}} \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\bar{\boldsymbol{\xi}}) \right|^2 \tag{192}$$

$$= \mathbb{E}_{\boldsymbol{\xi},\bar{\boldsymbol{\xi}} \sim \boldsymbol{\sigma}_d} \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \left| \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\bar{\boldsymbol{\xi}}) \right|^2 \tag{193}$$

$$= \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \mathbb{E}_{\boldsymbol{\xi},\bar{\boldsymbol{\xi}} \sim \boldsymbol{\sigma}_d} \left| \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}) \overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\bar{\boldsymbol{\xi}}) \right|^2 \tag{194}$$

$$= \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} 1 \tag{195}$$

$$= \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r}) \tag{196}$$

and

$$X_i = \mathbb{E}_{\boldsymbol{\xi}\sim\boldsymbol{\sigma}_d}\mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi}_i,\boldsymbol{\xi})^2 \tag{197}$$

$$= \mathbb{E}_{\boldsymbol{\xi}\sim\boldsymbol{\sigma}_d}\left|\sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda_{\mathcal{K}_{\text{Sym}}}\sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_i)\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})\right|^2 \tag{198}$$

$$= \mathbb{E}_{\boldsymbol{\xi}\sim\boldsymbol{\sigma}_d}\sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r})\sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}\left|\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_i)\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})\right|^2 \tag{199}$$

$$= \sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r})\sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}\mathbb{E}_{\boldsymbol{\xi}\sim\boldsymbol{\sigma}_d}\left|\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_i)\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi})\right|^2 \tag{200}$$

$$= \sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r})\sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}|\overline{\boldsymbol{Y}}^{\text{Sym}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_i)|^2 \tag{201}$$

$$= \sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r})\sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}\left(\frac{1}{w}\sum_u\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u})^2+\frac{1}{w}\sum_{u\neq v}\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u})\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,v})\right) \tag{202}$$

$$= \mathbb{E}_{\boldsymbol{\xi},\bar{\boldsymbol{\xi}}\sim\boldsymbol{\sigma}_d}\mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi},\bar{\boldsymbol{\xi}})^2+\sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r})\sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}\left(\frac{1}{w}\sum_{u\neq v}\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u})\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,v})\right) \tag{203}$$

Let

$$X_{\boldsymbol{r},i} = \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})}\left(\frac{1}{w}\sum_{u\neq v}\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u})\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,v})\right) \tag{204}$$

then

$$\Delta_i = \sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r}\lambda^2_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r})X_{\boldsymbol{r},i} \tag{205}$$

We replace the maximal function by the l^q-norm for $q\geq 1$,

$$\mathbb{E}\sup_{i\in[n(d)]}|\Delta_i|\leq\mathbb{E}(\sum_{i\in[n(d)]}|\Delta_i|^q)^{\frac{1}{q}}\leq(\mathbb{E}\sum_{i\in[n(d)]}|\Delta_i|^q)^{\frac{1}{q}}=n(d)^{\frac{1}{q}}(\mathbb{E}|\Delta_i|^q)^{\frac{1}{q}} \tag{206}$$

where the first three expectations are taken over $\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_{n(d)}\sim\boldsymbol{\sigma}_d$ and the last one is taken over $\boldsymbol{\xi}_i\sim\boldsymbol{\sigma}_d$. Then we replace the L^q-norm by the L^2-norm via Hypercontractivity, in which we used **Assumption Poly-ϕ** which implies that Δ_i is a polynomial of bounded degree

$$\mathbb{E}\sup_{i\in[n(d)]}|\Delta_i|\leq n(d)^{\frac{1}{q}}(\mathbb{E}|\Delta_i|^q)^{\frac{1}{q}}\leq C_q n(d)^{\frac{1}{q}}(\mathbb{E}|\Delta_i|^2)^{\frac{1}{2}}=C_q n(d)^{\frac{1}{q}}(\mathbb{E}\Delta_i^2)^{\frac{1}{2}} \tag{207}$$

We expand the L^2-norm and use orthogonality to "erase" the off-diagonal terms twice: first for $\boldsymbol{r}\neq\bar{\boldsymbol{r}}$

$$\mathbb{E}_{\boldsymbol{\xi}_i\sim\boldsymbol{\sigma}_d}X_{\boldsymbol{r},i}X_{\bar{\boldsymbol{r}},i}=0 \tag{208}$$

and second for $\boldsymbol{l} \neq \boldsymbol{l}'$ or $u \neq v$

$$\mathbb{E}_{\boldsymbol{\xi}_i \sim \boldsymbol{\sigma}_d} X_{\boldsymbol{r},i}^2 = \frac{1}{w^2} \mathbb{E}_{\boldsymbol{\xi}_i \sim \boldsymbol{\sigma}_d} \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} \left(\sum_{u \neq v} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,v}) \right) \sum_{\boldsymbol{l}'} \left(\sum_{u' \neq v'} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}'}(\boldsymbol{\xi}_{i,u'}) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}'}(\boldsymbol{\xi}_{i,v'}) \right) \tag{209}$$

$$= \frac{2}{w^2} \mathbb{E}_{\boldsymbol{\xi}_i \sim \boldsymbol{\sigma}_d} \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} \left(\sum_{u \neq v} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u})^2 \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,v})^2 \right) = \frac{2}{w^2} \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} w(w-1) \tag{210}$$

$$= \frac{2(w-1)}{w} \sum_{\boldsymbol{l}} 1 = \frac{2(w-1)}{w} \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r}) \leq 2\boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r}) \tag{211}$$

Combining this estimate with Eq. (196), Eq. (205) and Eq. (207) yields

$$(\mathbb{E}X_i)^{-1} (\mathbb{E} \sup_{i \in [n(d)]} |\Delta_i|) \leq C_q n(d)^{\frac{1}{q}} \frac{(2 \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}^4(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r}))^{1/2}}{\sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}^2(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} \tag{212}$$

$$\leq \sqrt{2} C_q n(d)^{\frac{1}{q}} \frac{\sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}^2(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})^{1/2}}{\sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}^2(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} \tag{213}$$

$$\leq \sqrt{2} C_q n(d)^{\frac{1}{q}} \sup_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \frac{\lambda_{\mathcal{K}_{\text{Sym}}}^2(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})^{1/2}}{\lambda_{\mathcal{K}_{\text{Sym}}}^2(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} \tag{214}$$

$$\sim 2C_q d^{\frac{r}{q}} \sup_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})^{-\frac{1}{2}} \tag{215}$$

$$\sim d^{\frac{r}{q}} \sup_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} d^{-|\boldsymbol{r}|\alpha_p/2} \xrightarrow[d \to \infty]{} 0 \tag{216}$$

by choosing q (independent of d) sufficiently large.

The proof of Eq. (231) is similar. Let X_i denote the random variable

$$X_i \equiv \mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_i) \quad \text{and} \quad \Delta_i \equiv X_i - \mathbb{E}X_i \tag{217}$$

and it suffices to prove

$$\frac{\mathbb{E} \sup_{i \in [n(d)]} |\Delta_i|}{\mathbb{E}X_i} \xrightarrow[d \to \infty]{} 0 \tag{218}$$

We have

$$\mathbb{E}X_i = \mathbb{E}_{\boldsymbol{\xi}_i \sim \boldsymbol{\sigma}_d} \mathcal{K}_{\text{Sym},>m(d)}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_i) \tag{219}$$

$$= \mathbb{E}_{\boldsymbol{\xi}_i \sim \boldsymbol{\sigma}_d} \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \sum_{\boldsymbol{l} \in \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r})} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}^{\text{Sym}}(\boldsymbol{\xi}_i) \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}^{\text{Sym}}(\boldsymbol{\xi}_i) \tag{220}$$

$$= \sum_{\boldsymbol{r}:\mathfrak{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \boldsymbol{N}(\boldsymbol{d}_{pen}, \boldsymbol{r}) \tag{221}$$

and

$$\Delta_i = X_i - \mathbb{E}X_i = w^{-1} \sum_{\boldsymbol{r}:\mathscr{L}(\boldsymbol{r})>r} \lambda_{\mathcal{K}_{\text{Sym}}}(\boldsymbol{r}) \sum_{\boldsymbol{l}\in\boldsymbol{N}(\boldsymbol{d}_{pen},\boldsymbol{r})} \sum_{u\neq v} \overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,u})\overline{\boldsymbol{Y}}_{\boldsymbol{r},\boldsymbol{l}}(\boldsymbol{\xi}_{i,v}),. \tag{222}$$

The remaining steps (replacing the maximal function by the l^q-norm, and then the L^q-norm by the L^2-norm using hypercontractivity, etc.) are similar to that of the proof of Eq. (230), which are omitted here.

Appendix I. Kernel Concentration, Hypercontractivity and Generalization from Mei-Misiakiewicz-Montanari

For convenience, we briefly recap the analytical results regarding generalization bounds of kernel machines from Mei et al. (2021a) Sec 3.

Let $(\boldsymbol{X}_d, \boldsymbol{\sigma}_d)$ be a probability space and $\mathscr{H}_d$ be a compact self-adjoint positive definite operator from $L^2(\boldsymbol{X}_d, \boldsymbol{\sigma}_d) \to L^2(\boldsymbol{X}_d, \boldsymbol{\sigma}_d)$. We assume $\mathscr{H}_d \in L^2(\boldsymbol{X}_d \times \boldsymbol{X}_d)$. Let $\{\psi_{d,j}\}$ and $\{\lambda_{d,j}\}$ be the eigenfunctions and eigenvalues associated to $\mathscr{H}_d$, i.e.

$$\mathscr{H}_d\psi_{d,j}(x) \equiv \int_{y\in\boldsymbol{X}_d} \mathscr{H}_d(x,y)\psi_{d,j}(y)\boldsymbol{\sigma}_d(y) = \lambda_{d,j}\psi_{d,j}(x). \tag{223}$$

We assume the eigenvalues are in non-ascending order, i.e. $\lambda_{d,j+1} \geq \lambda_{d,j} \geq 0$. Note that

$$\sum_j \lambda_{d,j}^2 = \|\mathscr{H}_d\|^2_{L^2(\boldsymbol{X}_d\times\boldsymbol{X}_d)} < \infty. \tag{224}$$

The associated reproducing kernel Hilbert space (RKHS) is defined to be functions $f \in L^2(\boldsymbol{X}_d, \boldsymbol{\sigma}_d)$ with $\|\mathscr{H}_d^{-\frac{1}{2}} f\|_{L^2(\boldsymbol{X}_d,\boldsymbol{\sigma}_d)} < \infty$. Given a finite training set $X \subseteq \boldsymbol{X}_d$ and observed labels $f(X) \in \mathbb{R}^{|X|}$, the regressor is an extension operator defined to be

$$\mathscr{R}_X f(x) = \mathscr{H}_d(x,X)\mathscr{H}_d(X,X)^{-1} f(X)\,. \tag{225}$$

Intuitively, when "$X \to \boldsymbol{X}_d$" in some sense, we expect the following

$$\mathscr{R}_X f(x) = \mathscr{H}_d(x,X)\mathscr{H}_d(X,X)^{-1} f(X) \to \mathscr{R}_{\boldsymbol{X}_d} f(x) = \mathscr{H}_d(\mathscr{H}_d^{-1} f)(x) = f(x), \tag{226}$$

namely, "$\mathscr{R}_X \to \boldsymbol{I}_{\boldsymbol{X}_d}$" in some sense.

Leveraging tools from the non-asymptotic analysis of random matrices (Vershynin, 2010), the work Mei et al. (2021a) provides a very nice answer to the above question in terms of the decay property of the eigenvalues $\{\lambda_{d,j}\}$ and the hypercontractivity property of the eigenfunctions $\{\psi_{d,j}\}$. They show that $\mathscr{R}_X$ is essentially a projection operator onto the low eigenspace under certain regularity assumptions on the operator $\mathscr{H}_d$. These assumptions are stated via the relationship between the number of (training) samples $n = n(d)$, the tail behavior of the eigenvalues with index $\geq m = m(d)$ and the tail behavior of the operator $\mathscr{H}_d$

$$\mathscr{H}_{d,>m(d)}(x,\bar{x}) \equiv \sum_{j>m(d)+1} \lambda_j\psi_j(x)\psi_j(\bar{x}) \tag{227}$$

as the "input dimension" d becomes sufficiently large.

Assumption 4. We say that the the sequence of operator $\{\mathscr{H}_d\}_{d\geq 1}$ satiesfies the Kernel Concentration Property (KCP) with respect to the sequence $\{n(d), m(d)\}_{d\geq 1}$ if there exsts a sequence of integers $\{u(d)\}_{d\geq 1}$ with $u(d) \geq m(d)$ such that the following holds

(a) (**Hypercontractivity.**) Let $D_{u(d)} = \text{span}\{\psi_j : 1 \leq j \leq u(d)\}$. Then for any fixed $q \geq 1$, and $C = C(q)$ such that for $\boldsymbol{f} \in D_{u(d)}$

$$\|\boldsymbol{f}\|_{L^q(\boldsymbol{X}_d, \boldsymbol{\sigma}_d)} \leq C\|\boldsymbol{f}\|_{L^2(\boldsymbol{X}_d, \boldsymbol{\sigma}_d)} \tag{228}$$

(b) (**Eigen-decay.**) There exists $\delta > 0$, such that, for all d large enough, for $l = 1$ and 2,

$$n(d)^{2+\delta} \leq \frac{(\sum_{j\geq u(d)+1} \lambda_{d,j}^l)^2}{\sum_{j\geq u(d)+1} \lambda_{d,j}^{2l}} \tag{229}$$

(c) (**Concentration of Diagonals.**) For $\{x_i\}_{i\in[n(d)]} \sim \boldsymbol{\sigma}_d^{n(d)}$, we have:

$$\frac{\sup_{i\in[n(d)]} \left|\mathbb{E}_{x\sim\boldsymbol{\sigma}_d} \mathscr{H}_{d,>m(d)}(x_i, x)^2 - \mathbb{E}_{x,\bar{x}\sim\boldsymbol{\sigma}_d} \mathscr{H}_{d,>m(d)}(x, \bar{x})^2\right|}{\mathbb{E}_{x,\bar{x}\sim\boldsymbol{\sigma}_d} \mathscr{H}_{d,>m(d)}(x, \bar{x})^2} \xrightarrow[d\to\infty]{\text{in Prob.}} 0 \tag{230}$$

$$\frac{\sup_{i\in[n(d)]} \left|\mathscr{H}_{d,>m(d)}(x_i, x_i) - \mathbb{E}_{x\sim\boldsymbol{\sigma}_d} \mathscr{H}_{d,>m(d)}(x, x)\right|}{\mathbb{E}_{x\sim\boldsymbol{\sigma}_d} \mathscr{H}_{d,>m(d)}(x, x)} \xrightarrow[d\to\infty]{\text{in Prob.}} 0 \tag{231}$$

where $c_d \to 0$ in probability as $d \to \infty$.

Assumption 5. Let $\mathscr{H}_d$ and $\{m(d), n(d)\}_{d\geq 1}$ be the same as above.

(a) For $l = 1$ and 2, there exists $\delta > 0$ such that

$$n(d)^{1+\delta} \leq \frac{1}{\lambda_{d,m(d)+1}^l} \sum_{k=\lambda_{m(d)+1}} \lambda_{d,k}^l \tag{232}$$

(b) There exists $\delta > 0$ such that

$$m(d) \leq n(d)^{1-\delta} \tag{233}$$

(c) (Spectral Gap.) There exists $\delta > 0$ such that

$$n(d)^{1-\delta} \geq \frac{1}{\lambda_{d,m(d)}} \sum_{k\geq m(d)+1} \lambda_{d,k} \tag{234}$$

Let $\mathcal{P}_{>k}$ (similarly for $\mathcal{P}_k$, $\mathcal{P}_{\leq k}$, etc.) denote the projection operator

$$\mathcal{P}_{>k} f = \sum_{j>k} \langle f, \psi_j\rangle \psi_j \tag{235}$$

Theorem 16 (Mei et al. (2021a)) *Assume $\mathcal{H}_d$ satisfy* **Assumptions 4** *and* **5**. *Let $\{f_d\}_{d\geq 1}$ be a sequence of functions and let $X \sim \boldsymbol{\sigma}_d^{n(d)}$. Then for every $\epsilon > 0$,*

$$\|\mathcal{R}_X(f_d) - f_d\|^2_{L^2(\boldsymbol{X}_d,\boldsymbol{\sigma}_d)} = \|\mathcal{P}_{>m(d)} f_d\|^2_{L^2(\boldsymbol{X}_d,\boldsymbol{\sigma}_d)} + c_{d,\epsilon}\|f_d\|^2_{L^{2+\epsilon}(\boldsymbol{X}_d,\boldsymbol{\sigma}_d)} \tag{236}$$

where $c_{d,\epsilon} \to 0$ in probability as $d \to \infty$.

The theorem says, $\mathcal{R}_X$ is essentially the projection operator $\mathcal{P}_{\leq m(d)}$ in the sense that when restricted to $L^{2+\epsilon}(\boldsymbol{X}_d, \boldsymbol{\sigma}_d)$,

$$\mathcal{R}_X = \mathcal{P}_{\leq m(d)} + \text{Error}_{d,\epsilon}\,. \tag{237}$$

Appendix J. Figure Zoo

J.1. MLPs: Depth is not equal to Hierarchy

We compare a one hidden layer MLP and a four hidden layer MLP in Fig. 7. Unlike CNNs, increasing the number of layers does not improve the performance of MLPs much for both NTK regression and SGD. This is consistent with a theoretical result from Bietti and Bach (2020), which says the NTKs of Relu MLPs are essentially the same for any depth.

J.2. ImageNet Plots

Appendix K. ARCHITECTURE SPECIFICATIONS

In this section, we provide more details of the architectures used in the experiments.

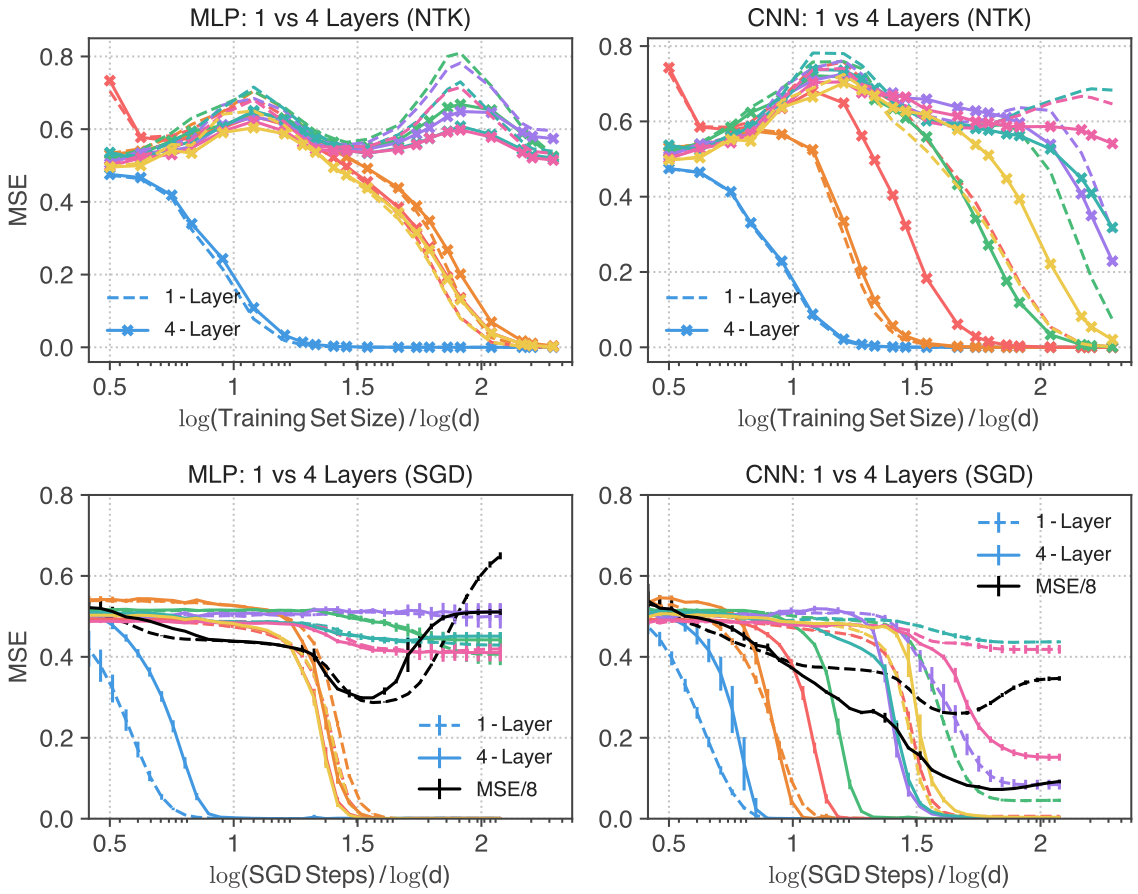


Figure 7: **MLPs do not benefit from having more layers.** We plot the learning dynamics vs training set size / SGD steps for each eigenfunction Y_i. Top: NTK regression and bottom: SGD + Momentum. Left: MLP; right: CNN. **Dashed lines / Solid lines** correspond to one-hidden / four-hidden layer networks. For both finite-width SGD training and infinite-width kernel regression, having more layers does not essentially improve performance of a MLP. This is in stark contrast to CNNs (right). By having more layers, the eigenstuctures of the kernels are refined.

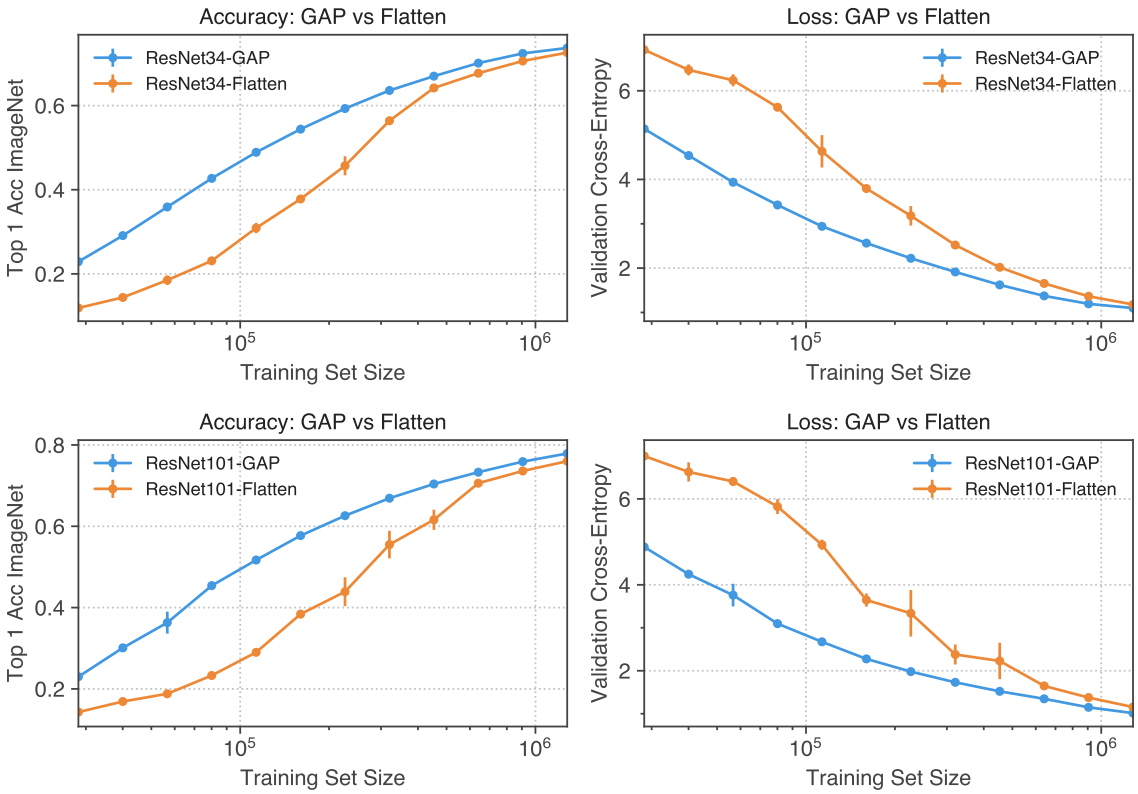


Figure 8: **ResNet-GAP vs ResNet-Flatten.** As the training set size increases, the performance (accuracy and loss) gap between the two shrinks substantially. Top/bottom ResNet34/ResNet101

```python
from neural_tangents import stax

def MLP(width=2048, depth=1, W_std=0.5, activation=stax.Rbf()):
    layers = []
    for _ in range(depth):
        layers += [stax.Dense(width, W_std=W_std), activation]
    layers += [stax.Dense(1)]
    return stax.serial(*layers)

def CNN(width=512, ksize=4, depth=1, W_std=0.5, activation=stax.Rbf(),
        readout=stax.Flatten(), act_after_readout=True):
    layers = []
    conv_op = stax.Conv(width, (ksize, 1), strides=(ksize, 1), W_std=W_std,
                        padding='VALID')
    for _ in range(depth):
        layers += [conv_op, activation]
    layers += [readout]
    if act_after_readout:
        layers += [stax.Dense(width * 4, W_std=W_std), activation]
    layers += [stax.Dense(1)]
    return stax.serial(*layers)

p = 4 # input shape: (-1, p**4, 1, 1)
S_MLP = MLP(depth=1) # Shallow MLP
D_MLP = MLP(depth=4) # Deep MLP

S_CNN = CNN(ksize=p, depth=1, act_after_readout=False) # Shallow CNN
# One layer CNN with an additional activation-dense layer
S_CNN_plus_Act = CNN(ksize=p, depth=1, act_after_readout=True)
D_CNN = CNN(ksize=p**2, depth=2) # Deep CNN
HR_CNN = CNN(ksize=p, depth=3) # High-resolution CNN
# High-resolution CNN with global average pooling readout
HR_CNN_GAP = CNN(ksize=p, depth=3, readout=stax.GlobalAvgPool(),
                 act_after_readout=False)
# High-resolution CNN with flattening as readout
HR_CNN_Flatten = CNN(ksize=p, depth=3, act_after_readout=False)
```

Listing 1: **Definitions of MLPs, S-CNN, D-CNN and HR-CNN used in the experiments**. The architectures used in the experiments of Fig. 3 are defined as follows. $\text{MLP}^{\otimes 4}$ = D_MLP, $\text{Conv}(p^2)^{\otimes 2}$ = D_CNN, $\text{Conv}(p)^{\otimes 4}$ = HR_CNN. The one layer MLP and one layer CNN used in Fig. 7 are S_MLP and S_CNN_plus_Act, resp. The $\text{Conv}(p)^{\otimes 3}$-Flatten $=$ HR_CNN_Flatten and $\text{Conv}(p)^{\otimes 3}$-GAP $=$ HR_CNN_GAP.