THE ONSET OF VARIANCE-LIMITED BEHAVIOR FOR NETWORKS IN THE LAZY AND RICH REGIMES

Anonymous authors
Paper under double-blind review

ABSTRACT

For small training set sizes P, the generalization error of wide neural networks is well-approximated by the error of an infinite width neural network (NN), either in the kernel or mean-field/feature learning regime. However, at a critical sample size P^*, the finite-width network generalization begins to worsen compared to the infinite width performance. In this work, we empirically study the transition from the infinite width behavior to this variance-limited regime as a function of sample size P and network width N. We find that finite size effects can become relevant for very small dataset sizes going as $P^* \sim \sqrt{N}$ for polynomial regression with ReLU networks. We discuss the source of this finite size behavior based on the variance of the NN's final neural tangent kernel (NTK). We then show how this transition can be pushed to larger P by enhancing feature learning or by ensemble averaging the network. We find that the learning curve for regression with the final NTK is an accurate approximation of the NN learning curve. Using this, we provide a toy model which also exhibits $P^* \sim \sqrt{N}$ scaling and has P-dependent benefits from feature learning.

1 INTRODUCTION

Deep learning systems are achieving state of the art performance on a variety of tasks (Tan & Le, 2019; Hoffmann et al., 2022). Exactly how their generalization is controlled by network architecture, training procedure, and task structure is still not fully understood. One promising direction for deep learning theory in recent years is the infinite width limit. Under a certain parameterization, infinite width networks yield a kernel method known as the neural tangent kernel (NTK) (Jacot et al., 2018; Lee et al., 2019). Kernel methods are easier to analyze, allowing for accurate prediction of the generalization performance for wide networks in this regime (Bordelon et al., 2020; Canatar et al., 2021; Bahri et al., 2021; Simon et al., 2021). Infinite width networks can also operate in the *mean-field* regime if network outputs are rescaled by a small parameter α that enhances feature learning (Mei et al., 2018; Chizat et al., 2019; Geiger et al., 2020b; Yang & Hu, 2020).

While infinite width networks provide useful limiting cases for deep learning theory, real networks have finite-width. Analysis at finite width is more difficult since predictions are dependent on the initialization of parameters. While several works have attempted to analyze feature evolution and kernel statistics at large but finite N (Dyer & Gur-Ari, 2020; Roberts et al., 2021), the implications of such finite size effects on generalization are not completely clear. Specifically, it is unknown at what value of sample size P finite width effects become relevant, what impact this has on the learning curve, and how it is affected by feature learning.

To identify the effect of finite width N and feature learning on the deviation from infinite width learning curves, we empirically study neural networks trained across a wide range of output scales α, widths N, and training set sizes P on the simple task of polynomial regression with a ReLU neural network. Concretely, our experiments show the following:

- Learning curves for polynomial regression transition exhibit significant finite width effects very early, around $P \sim \sqrt{N}$. Finite-width NNs at large α are always outperformed by their infinite width counterparts. We show this gap is driven primarily by variance of the predictor over initializations (Geiger et al., 2020a). Following prior work (Bahri et al., 2021), we

refer to this as the *variance-limited regime*. We compare three distinct ensembling methods to reduce error in this regime.

- NNs trained with feature learning show improved generalization both before and after the transition to the variance limited regime. Feature learning can be enhanced through rescaling the output of the network by a small scalar α or by training on a more complex task (a higher-degree polynomial). We show that alignment between the final NTK and the target function on test data improves with feature learning and sample size.
- We demonstrate that the learning curve for the NN is well captured by the learning curve for kernel regression with the *final* empirical NTK, eNTK$_f$, as has been observed in other works (Vyas et al., 2022; Geiger et al., 2020b; Atanasov et al., 2021; Wei et al., 2022).
- Using this correspondence between the NN and the final NTK, we provide a cursory account of how fluctuations in the final NTK over random initializations are suppressed at large width N and large feature learning strength. In a toy model, we reproduce several scaling phenomena, including the $P \sim \sqrt{N}$ transition and the improvements due to feature learning through an alignment effect.

1.1 Related Works

Geiger et al. (2020a) analyzed the scaling of network generalization with the number of model parameters. Since the NTK fluctuates with variance $O(N^{-1})$ for a width N network (Dyer & Gur-Ari, 2020; Roberts et al., 2021), they find that finite width networks in the lazy regime generically perform worse than their infinite width counterparts.

The scaling laws of networks over varying N and P were also studied, both empirically and theoretically by Bahri et al. (2021). They consider two types of learning curve scalings. First, they describe *resolution-limited* scaling, where either training set size or width are effectively infinite and the scaling behavior of generalization error with the other quantity is studied. Second, they analyze *variance-limited* scaling where width or training set size are fixed to a finite value and the other parameter is taken to infinity. While that work showed for any fixed P that the learning curve converges to the infinite width curve as $O(N^{-1})$, these asymptotics do not predict, for fixed N, at which value of P the NN learning curve begins to deviate from the infinite width theory. This is the focus of our work.

The contrast between rich and lazy networks has been empirically studied in several prior works. Depending on the structure of the task, the lazy regime can have either worse (Fort et al., 2020) or better (Ortiz-Jiménez et al., 2021; Geiger et al., 2020b) performance than the feature learning regime. For our setting, where the signal depends on only a small number of relevant input directions, we expect representation learning to be useful, as discussed in (Ghorbani et al., 2020; Paccolat et al., 2021b). Consequently, we posit and verify that the rich network will outperform the lazy one.

Our toy model is inspired by the literature on random feature models. Analysis of generalization for two layer networks at initialization in the limit of high dimensional data have been carried out using techniques from random matrix theory (Mei & Montanari, 2022; Hu & Lu, 2020; Adlam & Pennington, 2020a; Dhifallah & Lu, 2020; Adlam & Pennington, 2020b) and statistical mechanics (Gerace et al., 2020; d'Ascoli et al., 2020; d'Ascoli et al., 2020). Several of these works have identified that when N is comparable to P, the network generalization error has a contribution from variance over initial parameters. Further, they provide a theoretical explanation of the benefit of ensembling predictions of many networks trained with different initial parameters. Recently, Ba et al. (2022) studied regression with the hidden features of a two layer network after taking one step of gradient descent, finding significant improvements to the learning curve due to feature learning.

2 Problem Setup and Notation

We consider a supervised task with a dataset $\mathcal{D} = \{\boldsymbol{x}^\mu, y^\mu\}_{\mu=1}^P$ of size P. The pairs of data points are drawn from a population distribution $p(\boldsymbol{x}, y)$. Our experiments will focus on training networks to interpolate degree k polynomials on the sphere (full details in Appendix A). For this task, the infinite width network learning curves can be found analytically. In particular at large P the generalization error scales as $1/P^2$ (Bordelon et al., 2020). We take a single output feed-forward NN $\tilde{f}_\theta : \mathbb{R}^D \to \mathbb{R}$

with hidden width N for each layer. We let θ denote all trainable parameters of the network. Using NTK parameterization (Jacot et al., 2018), the activations for an input $\boldsymbol{x}$ are given by

$$h_i^{(\ell)} = \frac{\sigma}{\sqrt{N}} \sum_{j=1}^{N} W_{ij}^{(\ell)} \varphi(h_j^{(\ell-1)}), \quad \ell = 2, \dots L, \quad h_i^{(1)} = \frac{\sigma}{\sqrt{D}} \sum_{j=1}^{D} W_{ij}^{(1)} x_j. \tag{1}$$

Here, the output of the network is $\tilde{f}_\theta = h_1^{(L)}$. We will take φ to be the ReLU nonlinearity. At initialization we have $W_{ij} \sim \mathcal{N}(0, 1)$. Consequently, the scale of the output at initialization is $O(\sigma^L)$. We denote the scale of the output by $\alpha = \sigma^L$. α controls the feature learning strength of a given NN. Large α corresponds to a *lazy* network while small α yields a *rich* network with feature movement. More details on how α controls feature learning are given in Appendix C.1.

In what follows, we will denote the infinite width NTK limit of this network by NTK_∞. We will denote its finite width linearization by $\text{eNTK}_0(\boldsymbol{x}, \boldsymbol{x}') := \sum_\theta \partial_\theta f(\boldsymbol{x}) \partial_\theta f(\boldsymbol{x}')|_{\theta=\theta_0}$, and we will denote its linearization around its final parameters θ_f by $\text{eNTK}_f(\boldsymbol{x}, \boldsymbol{x}') := \sum_\theta \partial_\theta f(\boldsymbol{x}) \partial_\theta f(\boldsymbol{x}')|_{\theta=\theta_f}$.

Following other authors (Chizat et al., 2019; Adlam & Pennington, 2020a), we will take the output to be $f_\theta(\boldsymbol{x}) := \tilde{f}_\theta(\boldsymbol{x}) - \tilde{f}_{\theta_0}(\boldsymbol{x})$. Thus, at initialization the function output is 0. We explain this choice further in Appendix A. The parameters are then trained with full-batch gradient descent on a mean squared error loss. We denote the final network function starting from initialization θ_0 on a dataset $\mathcal{D}$ by $f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x})$ or f^* for short. The generalization error is calculated using a held-out test set and approximates the population risk $E_g(f) := \left\langle (f(\boldsymbol{x}) - y)^2 \right\rangle_{\boldsymbol{x}, y \sim p(\boldsymbol{x}, y)}$.

3 Empirical Results

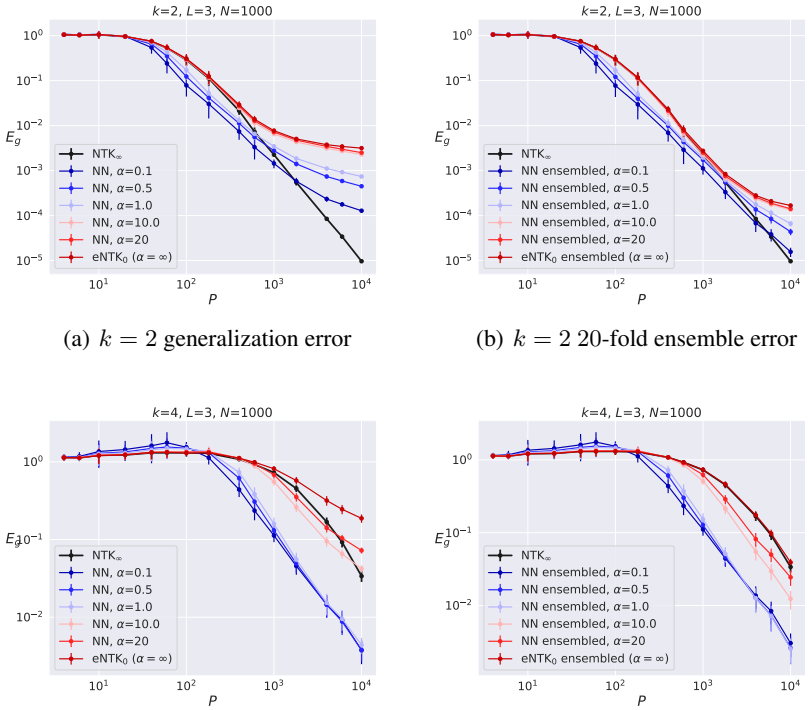
In this section, we will study learning curves for NNs trained on polynomial regression tasks of varying degrees. We will establish the following key observations, which we will set out to theoretically explain in Section 4.

1. Both eNTK_0 and sufficiently lazy networks perform strictly worse than NTK_∞ , but the ensembled predictors approach the NTK_∞ test error.
2. NNs in the feature learning regime of small α can outperform NTK_∞ for an intermediate range of P. Over this range, the effect of ensembling is less notable.
3. Even richly trained finite width NNs eventually perform worse than NTK_∞ at sufficiently large P. However, these small α feature-learning networks become variance-limited at larger P than lazy networks. Once in the variance-limited regime, all networks benefit from ensembling over initializations.
4. For all networks, the transition to the variance-limited regime begins at a P^* that scales sub-linearly with N. The scaling we find for the task of polynomial regression is close to $P^* \sim \sqrt{N}$.

3.1 Finite width effects cause the onset of a variance limited regime

In this section, we first investigate how finite width NN learning curves differ from infinite width NTK regression. In Figure 1 we show the generalization error $E_g(f^*_{\theta_0,\mathcal{D}})$ for a depth 3 network with width $N = 1000$ trained on a quadratic $k = 2$ and quartic $k = 4$ polynomial regression task. Additional plots for other degree polynomials are provided in Appendix E. We sweep over P to show the effect of more data on generalization, which is the main relationship we are interested in studying. For each training set size we sweep over a grid of 20 random draws of the train set and 20 random network initializations. This for 400 trained networks in total at each choice of P, k, N, α. We see that a discrepancy arises at large enough P where the neural networks begin to perform worse than NTK_∞.

We probe the source of the discrepancy between finite width NNs and NTK_∞ by ensemble averaging network predictions $\bar{f}_\mathcal{D}(\boldsymbol{x}) := \langle f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) \rangle_{\theta_0}$ over $E = 20$ initializations θ_0. In Figures 1b and 1d, we calculate the error of $\bar{f}_\mathcal{D}(\boldsymbol{x})$, each trained on the same dataset. We then plot $E_g(\bar{f}_\mathcal{D})$. This ensembled error approximates the bias in a bias-variance decomposition (Appendix B). Thus, any gap between 1 (a) and 1 (b) is driven by variance of $f_{\theta,\mathcal{D}}$ over θ.



(a) $k = 2$ generalization error

(b) $k = 2$ 20-fold ensemble error

(c) $k = 4$ generalization error

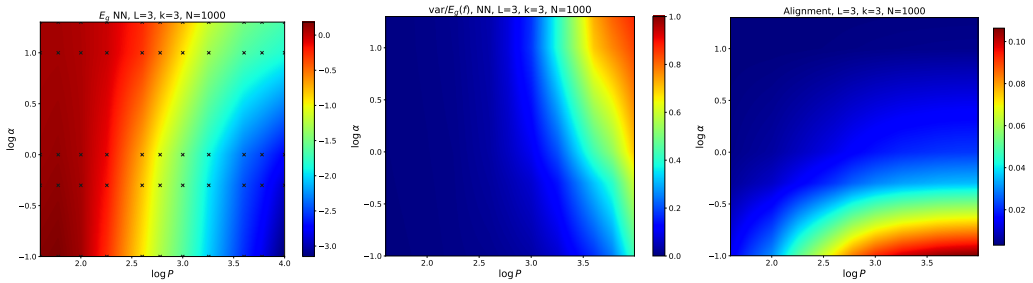
(d) $k = 4$ 20-fold ensemble error

Figure 1: Generalization errors of depth $L = 3$ neural networks across a range of α values compared to NTK_∞ . The regression for NTK_∞ was calculated using the Neural Tangents package (Novak et al., 2020). The exact scaling of NTK_∞ is known to go asymptotically as P^{-2} for this task. a) Lazy networks perform strictly worse than NTK_∞ while rich networks can outperform it for an intermediate range of P before their performance is also limited. b) Ensembling 20 networks substantially improves lazy network and eNTK_0 generalization, as well as asymptotic rich network generalization. This indicates that at sufficiently large P, these neural networks become limited by variance due to initialization. The error bars in a) and c) denote the variance due to both both training set and initialization. The error bars in b), d) denote the variance due to the train set.

We sharpen these observations with phase plots of NN generalization, variance and kernel alignment over P, α, as shown in Figure 2. In Figure 2a, generalization for NNs in the rich regime (small α) have lower final E_g than lazy networks. As the dataset grows, the fraction of E_g due to initialization variance (that is, the fraction removed by ensembling) strictly increases (2 (b)). We will show why this effect occurs in section 3.2. Figure 2b shows that, at any fixed P, the variance is lower for small α. To measure the impact of feature learning on the eNTK_f , we plot its alignment with the target function, measured as $\frac{\boldsymbol{y}^\top \boldsymbol{K} \boldsymbol{y}}{|\boldsymbol{y}|^2 \text{Tr} \boldsymbol{K}}$ for a test set of targets $[\boldsymbol{y}]_\mu$ and kernel $[\boldsymbol{K}]_{\mu\nu} = \text{eNTK}_f(\boldsymbol{x}_\mu, \boldsymbol{x}_\nu)$. Alignment of the kernel with the target function is known to be related to good generalization (Canatar et al., 2021). In Section 4, we revisit these effects in a simple model which relates kernel alignment and variance reduction.

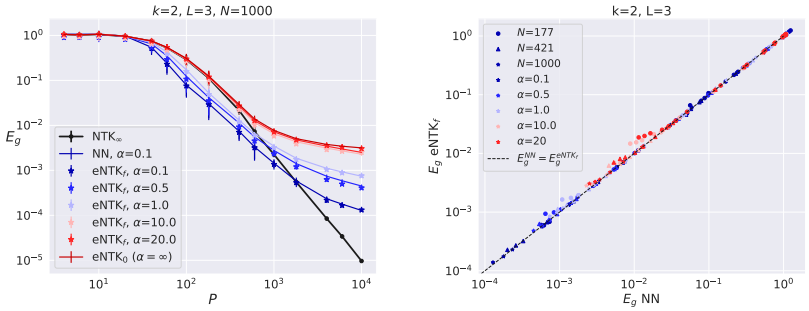
In addition to initialization variance, variance over dataset $\mathcal{D}$ contributes to the total generalization error. Following (Adlam & Pennington, 2020b), we discuss a symmetric decomposition of the variance in Appendix B, showing the contribution from dataset variance and the effects of bagging. We find that most of the variance in our experiments is due to initialization.

We show several other plots of the results of these studies in the appendix. We show the effect of bagging (Figure 7), phase plots of different degree target functions (Figures 8, 9), phase plots over N, α (Figure 10) and a comparison of network predictions against the initial and final kernel regressors (Figures 11, 12).



(a) $k = 3$ generalization error (b) $k = 3$ variance fraction (c) $k = 3$ alignment

Figure 2: Phase plots in the P, α plane of a) The log generalization error $\log_{10} E_g(f^\star)$, b) The fraction of generalization error removed by ensembling $1 - E_g(\bar{f}^\star)/E_g(f^\star)$, c) Kernel-task alignment measured by $\frac{\boldsymbol{y}^T K_f \boldsymbol{y}}{|\boldsymbol{y}|^2 \mathrm{Tr} K_f}$ where $\boldsymbol{y}$ and K_f are evaluated on test data. We have plotted 'x' markers in a) to show the points where the NNs were trained.



(a) $E_g^{NN} = E_g^{NTK_f}$ (b) $E_g^{NN} = E_g^{NTK_f}$ across N, α

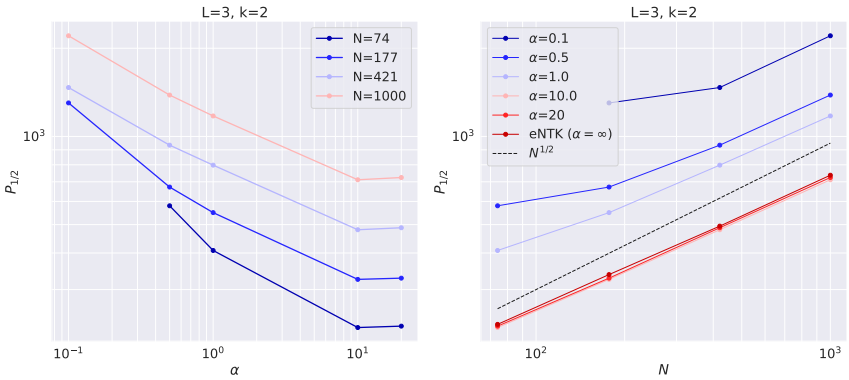
Figure 3: Performing kernel regression with the final NTK reproduces the learning curves of the neural network with high fidelity. (a) Learning curves across different laziness settings α in a width 1000 network. The solid black curve is the infinite width network. Solid colored curves are the neural network generalizations. Stars represent the final kernels, and lie on top of the corresponding neural network learning curves. (b) The agreement of generalizations between neural network and final kernel across different widths and feature learning strengths.

3.2 Final NTK Variance leads to Generalization Plateau

In this section, we show how the variance over initialization can be interpreted as kernel variance in both the rich and lazy regimes. We also show how this implies a plateau for the generalization error.

To begin, we demonstrate empirically that all networks have the same generalization error as kernel regression solutions with their *final* eNTKs. At large α, the initial and the final kernel are already close, so this follows from earlier results of Chizat et al. (2019). In the rich regime, the properties of the eNTK$_f$ have been studied in several prior works. Several have empirically demonstrated that the eNTK$_f$ is a good match to the final network predictor for a trained network (Long, 2021; Vyas et al., 2022; Wei et al., 2022) while others have given conditions under which such an effect would hold true (Atanasov et al., 2021; Bordelon & Pehlevan, 2022). We show in Figure 3 how the final network generalization error matches the generalization error of eNTK$_f$. As a consequence, we can use eNTK$_f$ to study the observed generalization behavior.

Next, we relate the variance of the final predictor $f^*_{\theta_0,\mathcal{D}}$ to the corresponding infinite width network $f^\infty_{\mathcal{D}}$. The finite size fluctuations of the kernel at initialization have been studied in (Dyer & Gur-Ari, 2020; Hanin & Nica, 2019; Roberts et al., 2021). The variance of the kernel elements has been shown to scale as $1/N$. We perform the following bias-variance decomposition: Take $f_{\theta_0,\mathcal{D}}$ to be



(a) Scaling of $P_{1/2}$ with α (b) Scaling of $P_{1/2}$ with N

Figure 4: Critical sample size $P_{1/2}$ measures the onset of the variance limited regime as a function of α at fixed N. (a) More feature learning (small α) delays transition to the variance limited regime. (b) $P_{1/2}$ as a function of N for fixed α has roughly $P_{1/2} \sim \sqrt{N}$ scaling.

the eNTK$_0$ predictor, or a sufficiently lazy network trained to interpolation on a dataset $\mathcal{D}$. Then,

$$\langle (f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) - y)^2 \rangle_{\theta_0,\mathcal{D},\boldsymbol{x},y} = \langle (f^\infty_{\mathcal{D}}(\boldsymbol{x}) - y)^2 \rangle_{\mathcal{D},\boldsymbol{x},y} + O(1/N). \tag{2}$$

We prove this equality using a relationship between the infinite-width network and an infinite ensemble of finite-width networks derived in Appendix B. There we also show that the $O(1/N)$ term is strictly positive for sufficiently large N. Thus, for lazy networks, finite width effects lead to strictly worse generalization error. The decomposition in Equation 2 continues to hold for rich networks at small α if f^∞ is interpreted as the infinite-width mean field limit. In this case one can show that ensembles of rich networks are approximating an infinite width limit in the mean-field regime. See Appendix B for details.

3.3 Feature Learning delays variance limited transition

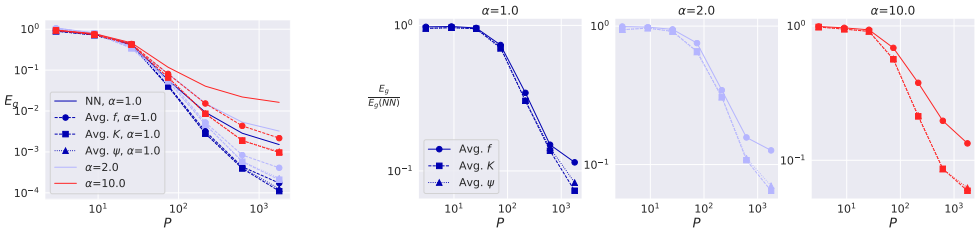
We now consider how feature learning alters the onset of the variance limited regime, and how this onset scales with α, N. We define the onset of the variance limited regime to take place at the value $P^* = P_{1/2}$ where over half of the generalization error is due to variance over initializations. Equivalently we have $E_g(\bar{f}^*)/E_g(f^*) = 1/2$. By using an interpolation method together with bisection, we solve for $P_{1/2}$ and plot it in Figure 4.

Figure 4b shows that $P_{1/2}$ scales as $\sqrt{N}$ for this task. In the next section, we shall show that this scaling is governed by the fact that $P_{1/2}$ is close to the value where the infinite width network generalization curve E_g^∞ is equal to the variance of the final kernel. In this case the quantities to compare are $E_g^\infty \approx P^{-2}$ and $\mathrm{Var}\,\mathrm{eNTK}_f \approx N^{-1}$.

We can understand the delay of the variance limited transition, as well as the lower value of the final plateau using a mechanistic picture similar to the effect observed in Atanasov et al. (2021). In that setting, under small initialization, the kernel follows a deterministic trajectory, picking up a low rank component in the direction of the train set targets $\boldsymbol{y}\boldsymbol{y}^\top$, and then changing only in scale as the network weights grow to interpolate the dataset. In their case, for initial output scale σ^L, the final kernel is deterministic up to a variance of $O(\sigma)$. In our case, the kernel variance at initialization scales as σ^{2L}/N. As $\sigma \to 0$ the kernel's trajectory becomes deterministic up to a variance term scaling with σ as $O(\sigma)$, which implies that the final predictor also has a variance scaling as $O(\sigma)$.

4 Signal plus noise correlated feature model

In Section 3.2 we have shown that in both the rich and lazy regimes, the generalization error of the NN is well approximated by the generalization of a kernel regression solution with eNTK$_f$. This



(a) Ensembling Methods

(b) Reduction in E_g for Each Ensembling Technique

Figure 5: The random feature model suggests three possible types of ensembling: averaging the output function $f(\boldsymbol{x}, \theta)$, averaging the final NTK $K(\boldsymbol{x}, \boldsymbol{x}'; \theta)$, and averaging the induced features $\boldsymbol{\psi}(\boldsymbol{x}, \theta)$. We analyze these ensembling methods for a $k = 1$ task with a width $N = 100$ ReLU network. (a) While all three ensembling methods improve generalization, averaging either the kernel $\langle K \rangle$ or features $\langle \psi \rangle$ gives a better improvement to generalization than averaging the output function $\langle f \rangle$. This indicates that computing final kernels for many richly trained networks and performing regression with this averaged kernel gives the best performance. (b) We plot the relative error of each ensembling method against the single init neural network. The gap between ensembling and the single init NN becomes evident for sufficiently large $P \sim P_{1/2}$. For small α, all ensembling methods perform comparably, while for large α ensembling the kernel or features gives much lower E_g than averaging the predictors.

finding motivates an analysis of the generalization of kernel machines which depend on network initialization θ_0. Unlike many analyses of random feature models which require high dimensional Gaussian random data, and focus on two layer networks (Mei & Montanari, 2022; Adlam & Pennington, 2020a; Gerace et al., 2020; Ba et al., 2022), we propose to analyze regression with the eNTK$_f$ for an arbitrary depth network on an arbitrary data distribution. This work builds on the kernel generalization theory for non-random kernels developed with statistical mechanics (Bordelon et al., 2020; Canatar et al., 2021; Simon et al., 2021). We will attempt to derive approximate learning curves in terms of the final NTK's signal and noise components, which provide some phenomenological explanations of the onset of the variance limited regime and the benefits of feature learning. Starting with the final NTK $K_{\theta_0}(\boldsymbol{x}, \boldsymbol{x}')$ which depends on the random initial parameters θ_0, we project its square root $K_{\theta_0}^{1/2}(\boldsymbol{x}, \boldsymbol{x}')$ on a fixed orthonormal (with respect to $p(\boldsymbol{x})$) basis $\{b_k(x)\}_{k=1}^{\infty}$ to define a feature map

$$\psi_k(\boldsymbol{x}, \theta) = \int d\boldsymbol{x}' p(\boldsymbol{x}') K_{\theta_0}^{1/2}(\boldsymbol{x}, \boldsymbol{x}') b_k(\boldsymbol{x}') \ , \ k \in \{1, ..., \infty\}. \tag{3}$$

The kernel can be reconstructed from these features $K_{\theta_0}(\boldsymbol{x}, \boldsymbol{x}') = \sum_k \psi_k(\boldsymbol{x}, \theta_0)\psi_k(\boldsymbol{x}', \theta_0)$. The kernel interpolation problem can be solved by performing linear regression with features $\boldsymbol{\psi}(\boldsymbol{x}, \theta_0)$. Here, $\boldsymbol{w}(\theta_0) = \lim_{\lambda \to 0} \text{argmin}_{\boldsymbol{w}} \sum_{\mu=1}^{P} [\boldsymbol{w} \cdot \boldsymbol{\psi}(\boldsymbol{x}_\mu, \theta_0) - y_\mu]^2 + \lambda |\boldsymbol{w}|^2$. The learned function $f(\boldsymbol{x}, \theta_0) = \boldsymbol{w}(\theta_0) \cdot \boldsymbol{\psi}(\boldsymbol{x}, \theta_0)$ is the minimum norm interpolator for the kernel $K(\boldsymbol{x}, \boldsymbol{x}'; \theta_0)$, and from 3 would match the neural network learning curve. Since the target function y does not depend on the initialization θ_0, we decompose it in terms of the mean features $\bar{\boldsymbol{\psi}}(\boldsymbol{x}) = \langle \boldsymbol{\psi}(\boldsymbol{x}, \theta_0) \rangle_{\theta_0}$ as $y(\boldsymbol{x}) = \boldsymbol{w}^* \cdot \bar{\boldsymbol{\psi}}(\boldsymbol{x})$. For convenience, we define the fluctuation of the feature as $\boldsymbol{\delta}(\boldsymbol{x}, \theta_0) = \boldsymbol{\psi}(\boldsymbol{x}, \theta_0) - \bar{\boldsymbol{\psi}}(\boldsymbol{x})$. The generalization error depends on signal $\boldsymbol{\Sigma}_s$ and noise $\boldsymbol{\Sigma}_n$ correlation matrices

$$\boldsymbol{\Sigma}_s = \left\langle \bar{\boldsymbol{\psi}}(\boldsymbol{x}) \bar{\boldsymbol{\psi}}(\boldsymbol{x})^\top \right\rangle_{\boldsymbol{x}} \ , \ \boldsymbol{\Sigma}_n = \left\langle \boldsymbol{\delta}(\boldsymbol{x}, \theta_0) \boldsymbol{\delta}(\boldsymbol{x}, \theta_0)^\top \right\rangle_{\boldsymbol{x}, \theta_0} . \tag{4}$$

Using the completeness of $\{b_k\}$, we decouple the initialization and sample dependence in the fluctuation $\boldsymbol{\delta}(\boldsymbol{x}, \theta_0) = \boldsymbol{\Sigma}_n^{1/2} \boldsymbol{A}(\theta_0) \boldsymbol{b}(\boldsymbol{x})$, where $\boldsymbol{A}$ is a mean zero with $\left\langle \boldsymbol{A}(\theta_0) \boldsymbol{A}(\theta_0)^\top \right\rangle_{\theta_0} = \boldsymbol{I}$.

In this random feature model, one can interpret the fluctuations in $K(\boldsymbol{x}, \boldsymbol{x}'; \theta_0)$ as generating fluctuations in the features $\boldsymbol{\psi}(\boldsymbol{x}, \theta_0)$ which induce fluctuations in the learned network predictor $f(\boldsymbol{x}, \theta_0)$. To illustrate the relative improvements to generalization from denoising these three different objects, in Figure 5, we compare

- averaging the final kernel K and performing kernel regression with $\langle K \rangle$,

- averaging the induced features ψ and performing linear regression with $\langle\psi\rangle$,
- averaging network predictions f directly.

For all α, all ensembling methods provide improvements over training a single NN. However, we find that averaging the kernel directly and performing regression with this kernel exhibits the largest reduction in generalization error. Averaging features performs comparably. However, ensemble averaging network predictors does not perform as well as either of these other two methods. The gap between these ensembling methods is more significant in the lazy regime (large α) and is negligible in the rich regime (small α).

4.1 Toy Model and Approximate Learning Curves

Working under the approximation that $\boldsymbol{A}(\theta_0)$'s entries are iid Gaussian over the disorder in θ_0, and assuming that P is much smaller than the total parameter count of the network, we obtain a modification the noise-free learning curves of Bordelon et al. (2020) and Canatar et al. (2021) which depend on $\{\boldsymbol{w}^*, \boldsymbol{\Sigma}_s, \boldsymbol{\Sigma}_n, P\}$. In this model, the generalization error will plateau to $E_g \sim \boldsymbol{w}^* \boldsymbol{\Sigma}_n [\boldsymbol{\Sigma}_n + \boldsymbol{\Sigma}_s]^{-1} \boldsymbol{\Sigma}_s \boldsymbol{w}^*$. Under the further assumption of codiagonalizable signal $\boldsymbol{\Sigma}_s = \sum_k \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top$ and noise $\boldsymbol{\Sigma}_n = \sum_k \eta_k \boldsymbol{u}_k \boldsymbol{u}_k^\top$ correlations, the estimation error along the k-th eigenfunction is

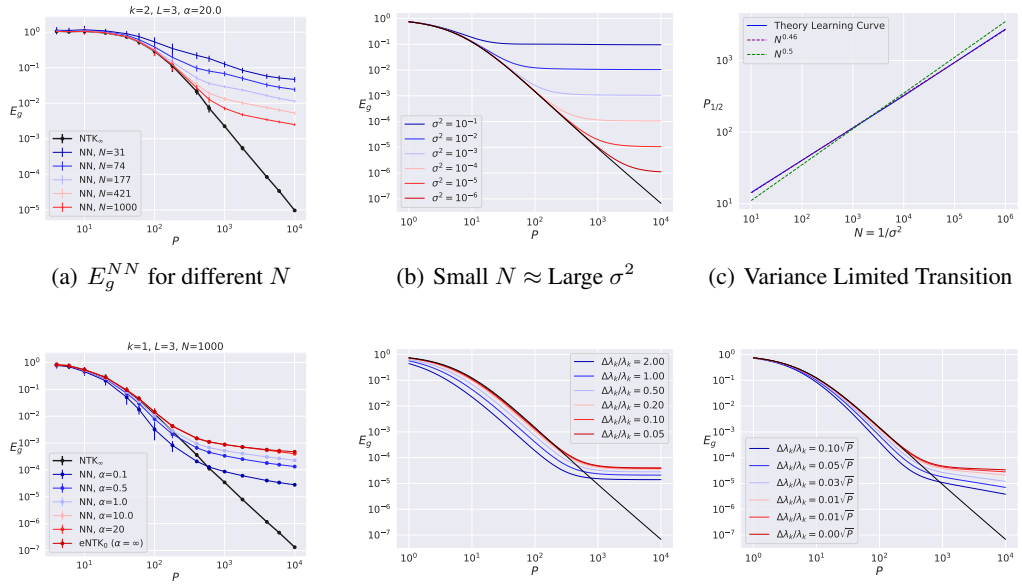
$$E_k(P) \sim \frac{1}{1 + P(\lambda_k + \eta_k)\kappa^{-1}} \left[\frac{1 + \eta_k P \kappa^{-1}}{1 + P\kappa^{-1}(\lambda_k + \eta_k)} + P\kappa^{-1}\eta_k \right], \tag{5}$$

where κ solves the implicit equation $1 = \sum_k \frac{\lambda_k + \eta_k}{(\lambda_k + \eta_k)P + \kappa}$. This k-th mode error plateaus to $\frac{\eta_k}{\lambda_k + \eta_k}$ which can be interpreted in terms of a signal to noise ratio $\frac{\lambda_k}{\eta_k}$. While the first term decays like P^{-1}, the second term approaches a constant at large P, giving the asymptotic error. Details of these calculations are given in Appendix D.

4.2 Explaining Feature Learning Benefits and Error Plateaus

Using this theory, we can attempt to explain some of the observed phenomena associated with the onset of the variance limited regime. First, we note that the noise correlations $\boldsymbol{\Sigma}_n$ should scale as $O(1/N)$ since kernels, either in the lazy or rich regime, exhibit variance $1/N$ fluctuations over initialization. In Figure 6 (a), we show learning curves for networks of different widths in the lazy regime. Small width networks enter the variance limited regime earlier and have higher error. Similarly, if we alter the scale of the noise in our toy model, the corresponding transition time $P_{1/2}$ is smaller and the asymptotic error is higher. In Figure 6 (c), we show that our theory also predicts the onset of the variance limited regime at $P_{1/2} \sim \sqrt{N}$. We again stress that this scaling is a consequence of the structure of the task. Since the target function is an eigenfunction of the kernel, the infinite width error goes as $1/P^2$. Since the variance scales as $1/N$, these two quantities become comparable at $P \sim \sqrt{N}$. Often, more complicated tasks exhibit power law decays where $E_g^{N=\infty} = P^{-\beta}$ with $\beta < 2$ (Spigler et al., 2020; Bahri et al., 2021). For such tasks, we expect a transition around $P_{1/2} \sim N^{1/\beta}$. For tasks that are harder to learn, namely those that have infinite-width generalization error scaling slower than $P^{-1/2}$, the variance onset can occur after the transition to the underparameterized regime. We leave it to future study to explore datasets with such scaling laws.

Using our model, we can also approximate the role of feature learning as enhancement in the signal correlation along task-relevant eigenfunctions. In Figure 6 (a) we plot the learning curves for networks trained with different levels of feature learning, controlled by α. We see that feature learning (small α) leads to improvements in the learning curve both before and after the variance limited regime begins. In Figure 6 (b), we plot the theoretical generalization for kernels with enhanced signal eigenvalue for the task eigenfunction $y(\boldsymbol{x}) = \phi_k(\boldsymbol{x})$. This enhancement, based on the intuition of kernel alignment, leads to lower bias and lower asymptotic variance. However, this model does not capture the fact that feature learning advantages are small at small P and that the slopes of the learning curves are different at different α. Following the observation of Paccolat et al. (2021a) that kernel alignment can occur with scale $\sqrt{P}$, we plot the learning curves for signal enhancements that scale as $\sqrt{P}$. This yields better qualitative agreement to the network's learning curve.



(a) E_g^{NN} for different N (b) Small $N \approx$ Large σ^2 (c) Variance Limited Transition

(d) Feature Scalings (e) Richness $\approx$ Amplified $\mathbf{\Sigma}_s$ (f) P-dependent Amplification

Figure 6: Our toy model of noisy features reproduces qualitative dependence of learning curves on kernel fluctuations and feature learning. (a) The empirical learning curves for networks of varying width N at large α. (b) Noisy kernel regression learning curve with noise eigenvalues $\eta_k = \sigma^2 \lambda_k$. The eigenvalues λ_k were taken from the infinite width NTK for our depth 3 ReLU network. The model predicts an asymptote at $E_k \sim \frac{1}{\sigma^2}$. (c) This toy model reproduces the approximate scaling of the transition sample size $P_{1/2} \sim N^{1/2}$. (d) Width $N = 1000$ networks trained with varying richness α. Small α improves the early learning curve and late stage learning curve. (e) Theory curves for a kernel with amplified eigenvalue $\lambda_k \to \lambda_k + \Delta\lambda_k$ corresponding to the target eigenfunction. This amplification mimics the effect of enhanced kernel alignment in the low α regime. Increasing this amplification factor improves the early generalization performance and the asymptote. (f) A model of P-dependent alignment where $\Delta\lambda_k \sim \sqrt{P}$ gives a better qualitative match to (d).

Though this toy model reproduces the onset of the variance limited regime $P_{1/2}$ and the reduction in variance due to feature learning, our current result is not the complete story. First, it is unclear if modeling $\boldsymbol{A}(\theta)$ as a Gaussian matrix is appropriate. For instance $\boldsymbol{A}$ could be zero off-diagonal and just have diagonal entries. Second, we assume that parameter count is much larger than P so our current equations would not capture the interesting effects at the interpolation threshold (Adlam & Pennington, 2020b). Lastly, in the lazy regime, the neural network learning curves do not asymptote as quickly as our model predicts. We leave these problems for future investigation.

5 Conclusion

We performed an extensive empirical study for deep ReLU NNs learning a fairly simple polynomial regression problems. For sufficiently large dataset size P, all neural networks under-perform the infinite width limit, and we demonstrated that this worse performance is driven almost entirely by initialization variance. We show that the onset of the variance limited regime can occur relatively early with scaling $P_{1/2} \sim \sqrt{N}$, but this can be delayed by enhancing feature learning. Finally, we have proposed a simple random-feature model to attempt to explain these effects and qualitatively reproduce the observed behavior, as well as quantitatively reproducing the relevant scaling relationship for $P_{1/2}$. This work takes a step towards understanding scaling laws in regimes where finite-size networks undergo feature learning. This has implications for how the choice of initialization scale, neural architecture, and number of ensembles of a network can be tuned to achieve optimal performance under a fixed compute and data budget.

REFERENCES

Ben Adlam and Jeffrey Pennington. The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization. In *International Conference on Machine Learning*, pp. 74–84. PMLR, 2020a.

Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-variance decomposition. *Advances in neural information processing systems*, 33:11022–11032, 2020b.

Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2021.

Jimmy Ba, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *arXiv preprint arXiv:2205.01445*, 2022.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws, 2021. URL `https://arxiv.org/abs/2102.06701`.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *arXiv preprint arXiv:2205.09653*, 2022.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1024–1034. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/bordelon20a.html`.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. *Version 0.2*, 5:14–24, 2018.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12, 2021.

Lénaïc Chizat, Edouard Oyallon, and Francis R. Bach. On lazy training in differentiable programming. In *NeurIPS*, 2019.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

Stéphane d'Ascoli, Levent Sagun, and Giulio Biroli. Triple descent and the two kinds of overfitting: Where & why do they appear? *Advances in Neural Information Processing Systems*, 33:3058–3069, 2020.

Oussama Dhifallah and Yue M Lu. A precise performance analysis of learning with random features. *arXiv preprint arXiv:2008.11904*, 2020.

Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1gFvANKDS`.

Stéphane d'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, pp. 2280–2290. PMLR, 2020.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5850–5861. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/405075699f065e43581f27d67bb68478-Paper.pdf`.

Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020a.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (11):113301, 2020b.

Federica Gerace, Bruno Loureiro, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Generalisation error in learning with random features and the hidden manifold model. In *International Conference on Machine Learning*, pp. 3452–3462. PMLR, 2020.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? In *NeurIPS*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/a9df2255ad642b923d95503b9a7958d8-Abstract.html`.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*, 2019.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hong Hu and Yue M Lu. Universality laws for high-dimensional learning with random features. *arXiv preprint arXiv:2009.07669*, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks (invited paper). *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2018.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jascha Sohl-Dickstein. Wide neural networks of any depth evolve as linear models under gradient descent. *ArXiv*, abs/1902.06720, 2019.

Philip M. Long. Properties of the after kernel. *CoRR*, abs/2105.10585, 2021. URL `https://arxiv.org/abs/2105.10585`.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

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`.

Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34:8998–9010, 2021.

Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001, apr 2021a. doi: 10.1088/1742-5468/abf1f3. URL `https://doi.org/10.1088/1742-5468/abf1f3`.

Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001, 2021b.

Daniel A. Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory, 2021.

James B. Simon, Madeline Dickens, and Michael R. DeWeese. Neural tangent kernel eigenvalues accurately predict generalization, 2021.

Stefano Spigler, Mario Geiger, and Matthieu Wyart. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124001, 2020.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Nikhil Vyas, Yamini Bansal, and Preetum Nakkiran. Limitations of the ntk for understanding generalization in deep learning. *arXiv preprint arXiv:2206.10012*, 2022.

Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. *arXiv preprint arXiv:2203.06176*, 2022.

Greg Yang and Edward J. Hu. Feature learning in infinite-width neural networks. *ArXiv*, abs/2011.14522, 2020.

A DETAILS ON EXPERIMENTS

We generated the dataset $\mathcal{D} = \{\boldsymbol{x}^\mu, y^\mu\}_{\mu=1}^P$ by sampling $\boldsymbol{x}^\mu$ uniformly on $\mathbb{S}^{D-1}$, the unit sphere in $\mathbb{R}^D$. $\tilde{y}$ was then generated as a Gegenbauer polynomial of degree k of a 1D projection of $\boldsymbol{x}$, $\tilde{y} = Q_k(\boldsymbol{\beta} \cdot \boldsymbol{x})$. Because the scale of the output of the neural network relative to the target is a central quantity in this work, it is especially important to make sure the target is appropriately scaled to unit norm. We did this by defining the target to be $y = \tilde{y}/\sqrt{\langle Q_k(\boldsymbol{\beta} \cdot \boldsymbol{x})^2 \rangle_{\boldsymbol{x} \sim \mathbb{S}^{D-1}}}$. The denominator can be easily and accurately approximated by Monte Carlo sampling.

We used JAX (Bradbury et al., 2018) for all neural network training. We built multi-layer perceptrons (MLPs) of depth 2 and 3. Most of the results are reported for depth 3 perceptrons, where there is a separation between the width of the network N and the number of parameters N^2. Sweeping over more depths and architectures is possible, but because of the extensive dimensionality of the hyperparameter search space, we have not yet experimented with deeper networks.

We considered MLPs with no bias terms. Since the Gegenbauer polynomials are mean zero, we do not need biases to fit the training set and generalize well. We have also verified that adding trainable biases does not change the final results in any substantial way.

As mentioned in the main text, we consider the final output function to be the initial network output minus the output at initialization:

$$f_\theta(\boldsymbol{x}) = \tilde{f}_\theta(\boldsymbol{x}) - \tilde{f}_{\theta_0}(\boldsymbol{x}). \tag{6}$$

Here, only θ is differentiated through, while θ_0 is held fixed. The rationale for this choice is that without this subtraction, in the lazy limit the trained neural network output can be written as

$$\tilde{f}^*_\theta(\boldsymbol{x}) = \tilde{f}_{\theta_0}(\boldsymbol{x}) + \sum_{\mu\nu} \boldsymbol{k}_\mu(\boldsymbol{x})[\boldsymbol{K}^{-1}]_{\mu\nu}(y^\nu - \tilde{f}_{\theta_0}(\boldsymbol{x})). \tag{7}$$

This is the same as doing eNTK$_0$ regression on the shifted targets $y^\mu - \tilde{f}_{\theta_0}(\boldsymbol{x})$. At large initialization the shift $\tilde{f}_{\theta_0}(\boldsymbol{x})$ amounts to adding random, initialization-dependent noise to the targets. By instead performing the subtraction, the lazy limit can be interpreted as a kernel regression on the targets themselves, which is preferable.

We trained this network with full batch gradient descent with a learning rate η so that

$$\begin{aligned} \Delta\theta &= -\eta \nabla_\theta \mathcal{L}(\mathcal{D}, \theta), \\ \mathcal{L}(\mathcal{D}, \theta) &:= \frac{1}{P} \sum_{\mu=1}^P |f_\theta(\boldsymbol{x}^\mu) - y^\mu|^2. \end{aligned} \tag{8}$$

Each network was trained to an interpolation threshold of 10^{-6}. If a network could not reach this threshold in under 30k steps, we checked if the training error was less than 10 times the generalization error. If this was not satisfied, then that run of the network was discarded.

For each fixed P, k, we generated 20 independent datasets. For each fixed N, α we generated 20 independent neural network initializations. This 20×20 table yields a total of 400 neural networks trained on every combination of initialization and dataset choice.

The infinite width network predictions were calculated using the Neural Tangents package (Novak et al., 2020). The finite width eNTK$_0$s were also calculated using the empirical methods in Neural Tangents. They were trained to interpolation using the `gradient_descent_mse` method. This is substantially faster than training the linearized model using standard full-batch gradient descent, which we have found to take a very long time for most networks. We use the same strategy for the eNTK$_f$s.

For the experiments in the main text, we have taken the input dimension to be $D = 10$ and sweep over $k = 1, 2, 3, 4$. We swept over 15 values P in logspace from size 30 to size 10k, and over 6 values of N in logspace from size 30 to size 2150. We then swept over alpha values $0.1, 0.5, 1.0, 10.0, 20.0$. Depending on α, N, we tuned the learning rate η of the network small enough to stay close to the gradient flow limit, but allow for the interpolation threshold to be feasibly reached.

For each of the 1800 settings of P, N, α, k and each of the 400 networks, 400 eNTK$_0$s, 400 eNTK$_f$s, and 20 NTK$_\infty$s, the generalization error was saved, as well as a vector of $\hat{y}$ predictions on a test set

of 2000 points. In addition, for the neural networks we saved both initial and final parameters. All are saved as lists of numpy arrays in a directory of about 1TB. We plan to make the results of our experiments publicly accessible, alongside the code to generate them.

B FINE-GRAINED BIAS-VARIANCE DECOMPOSITION

B.1 FINE GRAINED DECOMPOSITION OF GENERALIZATION ERROR

Let $\mathcal{D}$ be a dataset of $(\boldsymbol{x}^\mu, y^\mu)_{\mu=1}^P \sim p(\boldsymbol{x}, y)$ viewed as a random variable. Let θ_0 represent the inital parameters of a neural network, viewed as a random variable. In the case of no label noise, as in section 2.2.1 of Adlam & Pennington (2020b), we derive the symmetric decomposition of the generalization error in terms of the variance due to initialization and the variance due to the dataset. We have

$$\begin{aligned} E_g(f^*_{\theta_0,\mathcal{D}}) &= \langle (f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) - y)^2 \rangle_{\boldsymbol{x},y} = \langle (\langle f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) \rangle_{\theta_0,\mathcal{D}} - y)^2 \rangle_{\boldsymbol{x},y} + \mathbb{E}_{\boldsymbol{x}} \mathrm{Var}_{\theta_0,\mathcal{D}} f^*_{\theta_0,\mathcal{D}}(y) \\ &= \mathrm{Bias}^2 + V_{\mathcal{D}} + V_{\theta_0} + V_{\mathcal{D},\theta_0}. \end{aligned} \tag{9}$$

Here we have defined

$$\mathrm{Bias}^2 = \langle (\langle f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) \rangle_{\theta_0,\mathcal{D}} - y)^2 \rangle_{\boldsymbol{x},y}, \tag{10}$$

$$V_{\mathcal{D}} = \mathbb{E}_{\boldsymbol{x}} \, \mathrm{Var}_{\mathcal{D}} \, \mathbb{E}_{\theta_0}[f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) | \mathcal{D}] = \mathbb{E}_{\boldsymbol{x}} \, \mathrm{Var}_{\mathcal{D}} \, \bar{f}^*_{\mathcal{D}}(\boldsymbol{x}), \tag{11}$$

$$V_{\theta_0} = \mathbb{E}_{\boldsymbol{x}} \, \mathrm{Var}_{\theta_0} \, \mathbb{E}_{\mathcal{D}}[f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x}) | \theta_0], \tag{12}$$

$$V_{\mathcal{D},\theta_0} = \mathbb{E}_{\boldsymbol{x}} \mathrm{Var}_{\theta_0,\mathcal{D}} f^*_{\theta_0,\mathcal{D}}(y) - V_{\theta_0} - V_{\mathcal{D}}. \tag{13}$$

$V_{\mathcal{D}}$ and V_{θ_0} give the components of the variance explained by variance in $\mathcal{D}, \theta_0$ respectively. $V_{\mathcal{D},\theta_0}$ is the remaining part of the variance not explained by either of these two sources. As in the main text, $\bar{f}^*_{\mathcal{D}}(\boldsymbol{x})$ is the ensemble average of the trained predictors over initializations. $E_{\mathcal{D}}[f^*_{\theta_0,\mathcal{D}}(\boldsymbol{x})|\theta_0]$ is commonly referred to as the bagged predictor. In the next subsection we study these terms empirically.

B.2 EMPIRICAL STUDY OF DATASET VARIANCE

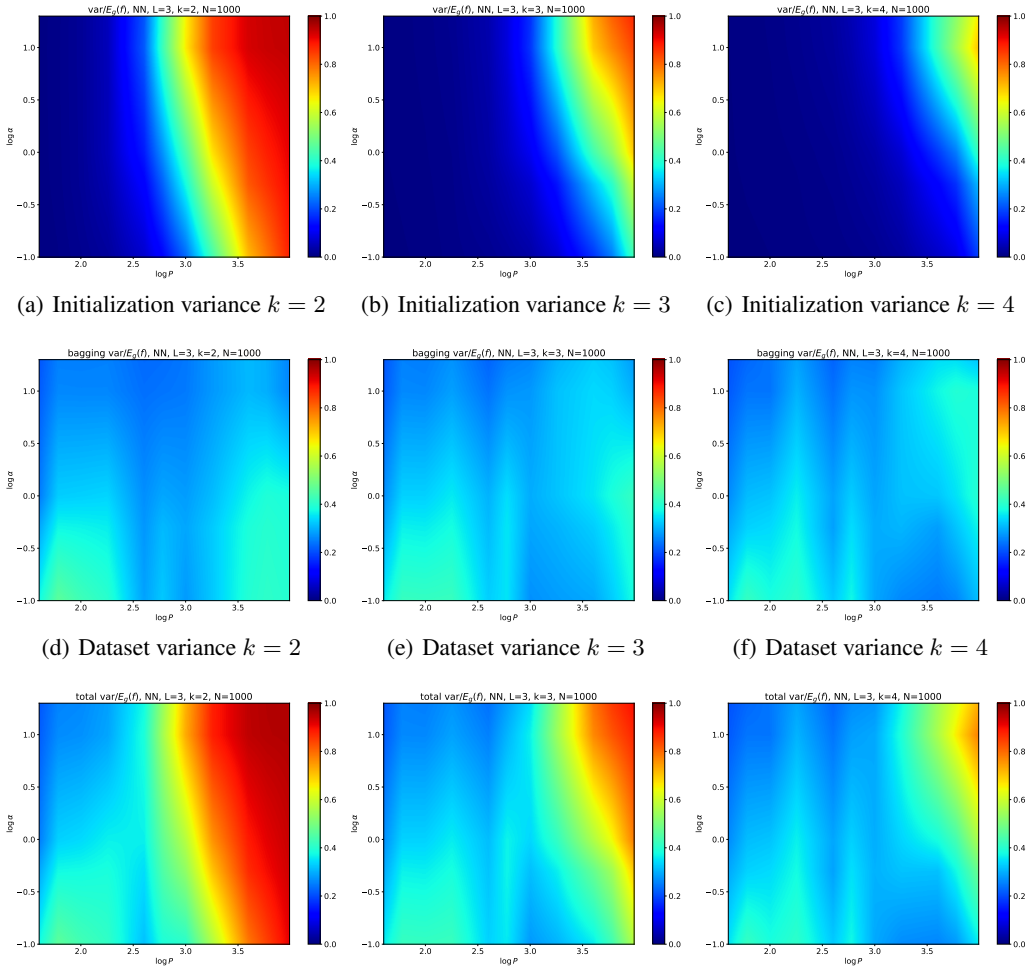
Using the network simulations, one can show that the bagged predictor does not have substantially lower generalization error in the regimes that we are interested in. This implies that most of the variance driving higher generalization error is due to variance over initializations. In figure 7, we make phase plots of the fraction of E_g that arises from variance due to initialization, variance over datasets, and total variance for width 1000. This can be obtained by computing the ensembled predictor, the bagged predictor, and the ensembled-bagged predictor respectively.

B.3 RELATING ENSEMBLED NETWORK GENERALIZATION TO INFINITE WIDTH GENERALIZATION

Making use of the fact that at leading order, the final NTK (either in the rich or lazy regime) of a trained network has θ_0-dependent fluctuations with variance $1/N$, one can write the kernel Gram matrices as

$$\begin{aligned} [\boldsymbol{K}_{\theta_0}]_{\mu\nu} &= [\boldsymbol{K}_\infty]_{\mu\nu} + \frac{1}{\sqrt{N}} [\boldsymbol{\delta K}_{\theta_0}]_{\mu\nu} + O(1/N) \\ [\boldsymbol{k}_{\theta_0}(\boldsymbol{x})]_\mu &= [\boldsymbol{k}_\infty(\boldsymbol{x})]_\mu + \frac{1}{\sqrt{N}} [\boldsymbol{\delta k}_{\theta_0}(\boldsymbol{x})]_\mu + O(1/N). \end{aligned} \tag{14}$$

Here, $\boldsymbol{\delta K}_{\theta_0}, \boldsymbol{\delta k}_{\theta_0}$ are the leading order fluctuations around the infinite width network. Because of how we have written them, their variance is $O(1)$ with respect to N. Using perturbation theory (Dyer & Gur-Ari, 2020), one can demonstrate that these leading order terms have mean zero around their infinite-width limit.



(a) Initialization variance $k = 2$ (b) Initialization variance $k = 3$ (c) Initialization variance $k = 4$

(d) Dataset variance $k = 2$ (e) Dataset variance $k = 3$ (f) Dataset variance $k = 4$

(g) Total variance $k = 2$ (h) Total variance $k = 3$ (i) Total variance $k = 4$

Figure 7: Phase plots of the fraction of the generalization error due to the initialization variance, the dataset variance, and their combined contribution. The columns correspond to the tasks of polynomial regression for degree $2, 3$ and 4 polynomials. Neural network has width 1000 and depth 3. Notice that the initialization variance dominates in the large P large α regime

The predictor for the eNTK$_0$ (or for a sufficiently large α neural network) for a training set with target labels $\boldsymbol{y}$ is given by:

$$\begin{aligned} f^*(\boldsymbol{x})_{\theta_0} &= \boldsymbol{k}_{\theta_0}(\boldsymbol{x})_\mu^\top \boldsymbol{K}_{\theta_0}^{-1} \cdot \boldsymbol{y} \\ &= f^\infty(\boldsymbol{x}) + \frac{1}{\sqrt{N}} \boldsymbol{\delta} \boldsymbol{k}_{\theta_0}(\boldsymbol{x})^\top \boldsymbol{K}_\infty^{-1} \cdot \boldsymbol{y} + \frac{1}{\sqrt{N}} \boldsymbol{k}_\infty(\boldsymbol{x})^\top \boldsymbol{K}_\infty^{-1} \boldsymbol{\delta} K_{\theta_0} \boldsymbol{K}_\infty^{-1} \cdot \boldsymbol{y} + O\left(N^{-1}\right). \end{aligned} \tag{15}$$

This implies (Geiger et al., 2020a):

$$\langle (f^*_{\theta_0}(\boldsymbol{x}) - f^\infty(\boldsymbol{x}))^2 \rangle_{\boldsymbol{x}} = O\left(N^{-1}\right). \tag{16}$$

Upon taking the ensemble, because of the mean zero property of the deviations, we get that

$$\begin{aligned} \langle f^*(\boldsymbol{x})_{\theta_0} \rangle_{\theta_0} &= f^\infty(\boldsymbol{x}) + O\left(N^{-1}\right) \\ &\Rightarrow \langle (\langle f^*(\boldsymbol{x})_{\theta_0} \rangle_{\theta_0} - f^\infty(\boldsymbol{x}))^2 \rangle = O\left(N^{-2}\right). \end{aligned} \tag{17}$$

We can now bound the generalization error of the ensemble of networks in terms of the infinite-width generalization:

$$\begin{aligned}\langle(\langle f^*_{\theta_0}(\boldsymbol{x})\rangle_{\theta_0} - y)^2\rangle_{\boldsymbol{x},y} = \langle(f^\infty(\boldsymbol{x}) - y)^2\rangle_{\boldsymbol{x},y} + \langle(\langle f^*_{\theta_0}(\boldsymbol{x})\rangle_{\theta_0} - f^\infty(\boldsymbol{x}))^2\rangle_{\boldsymbol{x}} \\ - 2\langle(f^\infty(\boldsymbol{x}) - y)(f^\infty(\boldsymbol{x}) - \langle f^*_{\theta_0}(\boldsymbol{x})\rangle_{\theta_0})\rangle_{\boldsymbol{x},y}.\end{aligned} \tag{18}$$

By equation 17, the second term yields a positive contribution going as $O(N^{-2})$. The last term can be bounded by Cauchy-Schwarz:

$$\begin{aligned}|\langle(f^\infty(\boldsymbol{x}) - y)(f^\infty(\boldsymbol{x}) - \langle f^*_{\theta_0}(\boldsymbol{x})\rangle_{\theta_0})\rangle_{\boldsymbol{x},y}| &\leq \sqrt{\langle(f^\infty(\boldsymbol{x}) - y)^2\rangle\langle(f^\infty(\boldsymbol{x}) - \langle f^*_{\theta_0}(\boldsymbol{x})\rangle_{\theta_0})^2\rangle_{\boldsymbol{x},y}} \\ &= \sqrt{E_g^\infty(P)c_1}/N.\end{aligned} \tag{19}$$

After we enter the variance limited regime by taking $P > P_{1/2}$ we get $E_g^\infty \leq O(1/N)$ so this last term is bounded by $N^{-3/2}$. Consequently, the difference in generalization error between the infinite width NTK and an ensemble of lazy network or eNTK$_0$ predictors is subleading in $1/N$ compared to the generalization gap, which goes as N^{-1}.

The same argument can be extended to any predictor that differs from some infinite width limit by fluctuations with variance N^{-2}. In particular Bordelon & Pehlevan (2022) show that the fluctuations of the final kernel in any mean field network are mean zero with variance N^{-1}. The above argument then applies to the predictor obtained by ensembling networks that have learned features. This implies that in the variance limited regime, ensemble averages of feature learning networks have the same generalization as the infinite-width mean field solutions up to a term that decays faster than $N^{-3/2}$.

C Feature Learning

C.1 Controlling Feature Learning Through Initialization Scale

Given the feed-forward network defined in equation 1, one can see that the components of the activations satisfy $h_i^{(\ell)} = O(\sigma h_i)^{(\ell-1)}$ and consequently that the output $h_1^{(L)} = O(\sigma^L)$. Because of the way the network is parameterized, the changes in the output $\frac{\partial f}{\partial \theta}$ also scale as $O(\sigma^L)$. This implies that the eNTK at any given time scales as

$$K_\theta(\boldsymbol{x}, \boldsymbol{x}') = \sum_\theta \frac{\partial f(\boldsymbol{x})}{\partial \theta}\frac{\partial f(\boldsymbol{x}')}{\partial \theta} = O(\sigma^{2L}). \tag{20}$$

After appropriately rescaling learning rate to $\eta = \sigma^{-2L}$ we get

$$\frac{df(\boldsymbol{x})}{dt} = -\eta \sum_\mu K_\theta(\boldsymbol{x}, \boldsymbol{x}^\mu)(f(\boldsymbol{x}^\mu) - y^\mu) \tag{21}$$

Under the assumption that $\sigma^L \ll 1$ and $y^\mu = O(1)$ so that the error term is $O(1)$ we get that the output changes in time as $df/dt = O(1)$.

On the other hand, using the chain rule one can show that the features change as a product of the gradient update and the features in the prior layer, yielding the scaling

$$\frac{dh^{(\ell)}}{dt} = \eta\frac{\sigma^L}{\sqrt{N}} = \frac{1}{\sigma^L\sqrt{N}} = (\alpha\sqrt{N})^{-1}. \tag{22}$$

This gives us that the change in the features scales as $(\alpha\sqrt{N})^{-1}$ while the change in the output scales as $O(1)$. Thus, for $\alpha\sqrt{N}$ sufficiently small, the features can move dramatically.

C.2 Kernel Alignment

In this section we comment on our choice of kernel alignment metric

$$A(\boldsymbol{K}) := \frac{\boldsymbol{y}^\top \boldsymbol{K}\boldsymbol{y}}{\operatorname{Tr}\boldsymbol{K}|\boldsymbol{y}|^2}. \tag{23}$$

For kernels that are diagonally dominant, such as those encountered in the experiments, this metric is related to another alignment metric

$$A_F(\boldsymbol{K}) := \frac{\boldsymbol{y}^\top \boldsymbol{K} \boldsymbol{y}}{|\boldsymbol{K}|_F |\boldsymbol{y}|^2}. \tag{24}$$

Here $|\boldsymbol{K}|_F$ is the Frobenius norm of the Gram matrix of the kernel. The advantage of the first metric over the second is that one can quickly estimate the denominator of $A(\boldsymbol{K})$ via Monte Carlo estimation of $\langle \boldsymbol{u}^\top \boldsymbol{K} \boldsymbol{u} \rangle_{\boldsymbol{u} \sim \mathcal{N}(0, \mathbf{1})}$.

We use $A(\boldsymbol{K}_f)$ as a measure of feature learning, as we have found that this more finely captures elements of feature learning than other related metrics. We list several metrics we tried that did not work.

One option for a representation-learning metric involves measuring the magnitude of the change between the initial and final kernels, $\boldsymbol{K}_i, \boldsymbol{K}_f$:

$$\Delta \boldsymbol{K} := |\boldsymbol{K}_f - \boldsymbol{K}_i|_F. \tag{25}$$

However, this is more sensitive to the raw parameter change than any task-relevant data. If one instead were to normalize the kernels to be unit norm at the beginning and the end, the modified metric

$$\Delta \boldsymbol{K} := \left| \frac{\boldsymbol{K}_f}{|\boldsymbol{K}_f|_F} - \frac{\boldsymbol{K}_i}{|\boldsymbol{K}_i|_F} \right|_F. \tag{26}$$

This metric however remains remarkably flat over the whole range of α, P, as does the centered kernel alignment (CKA) of Cortes et al. (2012)

$$\mathrm{CKA}(\boldsymbol{K}_i, \boldsymbol{K}_f) = \frac{\mathrm{Tr}[\boldsymbol{K}_i^c \boldsymbol{K}_f^c]}{\sqrt{|\boldsymbol{K}_i^c|_F |\boldsymbol{K}_f^c|_F}}, \quad \boldsymbol{K}^c = \boldsymbol{C}\boldsymbol{K}\boldsymbol{C}, \quad \boldsymbol{C} = \mathbf{1} - \frac{1}{P} \vec{1} \vec{1}^T. \tag{27}$$

Here C is the centering matrix that subtracts off the mean components of the kernel for a $P \times P$ kernel. This alignment metric has been shown to be useful in comparing neural representations (Kornblith et al., 2019). For our task, however, because the signal is low-dimensional, only a small set of eigenspaces of the kernel align to this task. As a result, the CKA, which counts all eigenspaces equally, appears to be too coarse to capture the low-dimensional feature learning that is happening.

On the other hand, we find that $A(\boldsymbol{K}_f)$ (with $\boldsymbol{K}_f$ given by the final kernel evaluated on a test set) can very finely detect alignment along the task relevant directions. This produces a clear signal of feature learning at small α and large P as shown in Figure 2c.

$A(\boldsymbol{K}_f)$ can be related to the centered kernel alignment between the final kernel and the (mean zero) task kernel $\boldsymbol{y}\boldsymbol{y}^\top$, where $\boldsymbol{y}$ is a vector of draws from the population distribution $p(\boldsymbol{x}, y)$.

D Generic Random Feature Model

D.1 Setting up the Problem

For a random kernel, $K(\boldsymbol{x}, \boldsymbol{x}'; \theta)$, we first compute its Mercer decomposition

$$\int d\boldsymbol{x}\, p(\boldsymbol{x}) K(\boldsymbol{x}, \boldsymbol{x}'; \theta) \phi_k(\boldsymbol{x}) = \lambda_k \phi_k(\boldsymbol{x}'). \tag{28}$$

From the eigenvalues and eigenfunctions, we can construct the square root

$$K^{1/2}(\boldsymbol{x}, \boldsymbol{x}'; \theta) = \sum_k \sqrt{\lambda_k} \phi_k(\boldsymbol{x}) \phi_k(\boldsymbol{x}'). \tag{29}$$

Lastly, using $K^{1/2}$, we can get a feature map by projecting against a static basis $\{b_k\}$ so $\psi_k(\boldsymbol{x}) = \int d\boldsymbol{x}' p(\boldsymbol{x}') K^{1/2}(\boldsymbol{x}, \boldsymbol{x}'; \theta)$. We now decompose these features into mean and variance $\boldsymbol{\psi}(\boldsymbol{x}, \theta) = \langle \boldsymbol{\psi}(\boldsymbol{x}, \theta) \rangle_\theta + \delta(\boldsymbol{x}, \theta)$. We note that since $b_k(\boldsymbol{x})$ form a complete orthonormal set of functions, that we can expand the fluctuation $\delta(\boldsymbol{x}, \theta)$ in the basis of b_k so

$$\boldsymbol{\delta}(\boldsymbol{x}, \theta) = \boldsymbol{\Sigma}_n^{1/2} \boldsymbol{A}(\theta)^\top \boldsymbol{b}(\boldsymbol{x}). \tag{30}$$

Now, the matrix $\boldsymbol{A}(\theta)$ depends on the random initialization θ, has mean zero over the disorder in θ and has correlation $\left\langle \boldsymbol{A}(\theta)\boldsymbol{A}(\theta)^\top \right\rangle_{\boldsymbol{\theta}} = \boldsymbol{I}$. We are interested in the minimizers of the regularized risk for dataset $\mathcal{D}$ and initialization θ

$$H(\boldsymbol{w};\mathcal{D},\theta) = \sum_{\mu=1}^{P} [\boldsymbol{w}\cdot\boldsymbol{\psi}(\boldsymbol{x}_\mu,\theta) - y_\mu]^2 + \lambda|\boldsymbol{w}|^2. \tag{31}$$

We decompose the target function in terms of the mean features $y(\boldsymbol{x}) = \boldsymbol{w}^*\boldsymbol{\Sigma}_s^{1/2}\boldsymbol{b}(\boldsymbol{x})$. The generalization error is

$$\begin{aligned} E_g(\mathcal{D},\theta) &= \left\langle \left[(\boldsymbol{w}-\boldsymbol{w}^*)\boldsymbol{\Sigma}_s^{1/2}\boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{w}^\top\boldsymbol{\Sigma}_n^{1/2}\boldsymbol{A}(\theta)^\top\boldsymbol{b}(\boldsymbol{x})\right]^2 \right\rangle \\ &= |\boldsymbol{\Sigma}_s^{1/2}(\boldsymbol{w}-\boldsymbol{w}^*) + \boldsymbol{A}(\theta)\boldsymbol{\Sigma}_n^{1/2}\boldsymbol{w}|^2, \end{aligned} \tag{32}$$

where $\boldsymbol{w}$ must be regarded as a function of $\mathcal{D},\theta$.

D.1.1 Replica Calculation Setup

We consider the performance of the model over random datasets $\mathcal{D}$ and random realizations of the initialization θ. We let the dimension of the RKHS be $N_{\mathcal{H}}$, which is distinct from the finite network width. We will work with the $P \ll N_{\mathcal{H}}$ limit since in our setting the RKHS dimension will be the total parameters in the neural network model which is much larger than the sample size. To make this computation tractable, we treat $\boldsymbol{A} \in \mathbb{R}^{N_{\mathcal{H}}\times N_{\mathcal{H}}}$ as a random Gaussian matrix with entries $A_{ij} \sim \mathcal{N}(0, N_{\mathcal{H}}^{-1})$ and let $\boldsymbol{b}_\mu \sim \mathcal{N}(0,\boldsymbol{I})$. Then,

$$Z[\mathcal{D},\theta,J] = \int d\boldsymbol{w} \exp\left(-\frac{\beta}{2}H(\boldsymbol{w};\mathcal{D},\theta) - \frac{\beta J}{2}E_g(\boldsymbol{w};\mathcal{D},\theta)\right). \tag{33}$$

in the large β limit. We note that the average generalization error can be obtained from $\langle \ln Z\rangle$

$$\langle E_g(\mathcal{D},\theta)\rangle = -\lim_{\beta\to\infty}\frac{2}{\beta}\frac{\partial}{\partial J}\langle \ln Z\rangle. \tag{34}$$

Based on intuitions from the physics of disordered systems, we expect $\ln Z$ to be self-averaging at large dataset and large feature dimension. We perform this average with the replica trick $\langle \ln Z\rangle = \lim_{n\to 0}\frac{1}{n}\ln\langle Z^n\rangle$, yielding

$$\langle Z^n\rangle = \int \prod_{a=1}^{n} d\boldsymbol{w}^a \mathbb{E}_{\mathcal{D},\theta} \exp\left(-\frac{\beta}{2\lambda}\sum_{a=1}^{n} H(\boldsymbol{w}^a;\mathcal{D},\theta) - \frac{\beta J}{2}\sum_{a=1}^{n} E_g(\boldsymbol{w}^a;\mathcal{D},\theta)\right). \tag{35}$$

We introduce the fields $\boldsymbol{\chi}^a = \boldsymbol{A}(\theta)\boldsymbol{\Sigma}_n^{1/2}\boldsymbol{w}^a$ which appear in both the train loss H and the test loss E_g. Over the Gaussian disorder in $\boldsymbol{A}$ these fields are jointly Gaussian with mean zero and covariance

$$\left\langle \chi_i^a \chi_j^b \right\rangle = \sum_{k,\ell}\langle A_{ik}A_{j\ell}\rangle\,[\boldsymbol{\Sigma}_n^{1/2}\boldsymbol{w}^a]_k[\boldsymbol{\Sigma}_n^{1/2}\boldsymbol{w}^a]_\ell = \frac{1}{N_{\mathcal{H}}}\delta_{ij}M_{ab}\ ,\ M_{ab} = \boldsymbol{w}^a\boldsymbol{\Sigma}_n\boldsymbol{w}^b. \tag{36}$$

The average over random data $\boldsymbol{b}_\mu$ factorize over data points by independence of samples, giving

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{b}} \exp\left(-\frac{\beta}{2\lambda}\sum_a [\boldsymbol{b}\cdot(\boldsymbol{\Sigma}_s^{1/2}(\boldsymbol{w}^a-\boldsymbol{w}^*) + \boldsymbol{\chi}^a)]^2\right) \\ &\propto \exp\left(-\frac{1}{2}\ln\det\left(\lambda\boldsymbol{I} + \beta\boldsymbol{Q} + \beta\boldsymbol{R} + \beta\boldsymbol{R}^\top + \beta\boldsymbol{M}\right)\right) \end{aligned} \tag{37}$$

where we introduced order parameters

$$Q_{ab} = (\boldsymbol{w}^a-\boldsymbol{w}^*)\boldsymbol{\Sigma}_s(\boldsymbol{w}^b-\boldsymbol{w}^*)\ ,\ R_{ab} = (\boldsymbol{w}^a-\boldsymbol{w}^*)\boldsymbol{\Sigma}_s^{1/2}\boldsymbol{\chi}^b. \tag{38}$$

We thus have

$$\begin{aligned}\langle Z^n\rangle = \int \prod_a d\boldsymbol{w}^a d\boldsymbol{Q} d\hat{\boldsymbol{Q}} d\boldsymbol{R} d\hat{\boldsymbol{R}} d\boldsymbol{M} d\hat{\boldsymbol{M}} \\ \exp\left(-\frac{P}{2}\ln\det\left(\lambda\boldsymbol{I}+\beta(\boldsymbol{Q}+\boldsymbol{R}+\boldsymbol{R}^\top+\boldsymbol{M})\right)-\frac{\beta}{2}\sum_a|\boldsymbol{w}^a|^2\right) \\ \times\exp\left(\frac{1}{2}\mathrm{Tr}[\boldsymbol{Q}\hat{\boldsymbol{Q}}+2\hat{\boldsymbol{R}}\boldsymbol{R}+\hat{\boldsymbol{M}}\boldsymbol{M}]\right) \\ \times\exp\left(-\frac{1}{2}(\boldsymbol{W}-\boldsymbol{w}^*\otimes\mathbf{1})[\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}](\boldsymbol{W}-\boldsymbol{w}^*\otimes\mathbf{1})-\frac{1}{2}\boldsymbol{W}[\boldsymbol{\Sigma}_n\otimes\hat{\boldsymbol{M}}]\boldsymbol{W}\right) \\ \times\mathbb{E}_{\{\boldsymbol{\chi}^a\}}\exp\left(-\sum_{ab}\hat{R}_{ab}(\boldsymbol{w}^a-\boldsymbol{w}^*)\boldsymbol{\Sigma}_s^{1/2}\boldsymbol{\chi}^b\right)\exp\left(-\frac{\beta J}{2}\mathrm{Tr}[\boldsymbol{Q}+2\boldsymbol{R}+\boldsymbol{M}]\right),\end{aligned}$$

where we introduced the shorthand $\boldsymbol{W} = \mathrm{Vec}\{\boldsymbol{w}^a\}_{a=1}^n$. The average over Gaussian $\boldsymbol{\chi}^a$ fields is now completely isolated in the $\hat{R}_{ab}$ terms.

$$\mathbb{E}_{\{\boldsymbol{\chi}^a\}}\exp\left(-\sum_{ab}\hat{R}_{ab}(\boldsymbol{w}^a-\boldsymbol{w}^*)\boldsymbol{\Sigma}_s^{1/2}\boldsymbol{\chi}^b\right)=\exp\left(\frac{1}{2N_{\mathcal{H}}}\mathrm{Tr}\hat{\boldsymbol{R}}^\top\boldsymbol{M}\hat{\boldsymbol{R}}\boldsymbol{Q}\right). \tag{39}$$

Lastly, we integrate over $\boldsymbol{W}$, giving

$$\begin{aligned}&\exp\left(-\frac{1}{2}\boldsymbol{w}^*\boldsymbol{\Sigma}_s\boldsymbol{w}^*\mathbf{1}^\top\hat{\boldsymbol{Q}}\mathbf{1}\right)\int d\boldsymbol{W}\exp\left(-\frac{1}{2}\boldsymbol{W}[\beta\boldsymbol{I}+\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}+\boldsymbol{\Sigma}_n\otimes\hat{\boldsymbol{M}}]\boldsymbol{W}\right) \\ &\quad\times\exp\left(\boldsymbol{W}[\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}](\boldsymbol{w}^*\otimes\mathbf{1})\right) \\ &=\exp\left(-\frac{1}{2}\boldsymbol{w}^*\boldsymbol{\Sigma}_s\boldsymbol{w}^*\mathbf{1}^\top\hat{\boldsymbol{Q}}\mathbf{1}\right) \qquad (40) \\ &\quad\times\exp\left(\frac{1}{2}(\boldsymbol{w}^*\otimes\mathbf{1})[\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}][\beta\boldsymbol{I}+\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}+\boldsymbol{\Sigma}_n\otimes\hat{\boldsymbol{M}}]^{-1}[\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}](\boldsymbol{w}^*\otimes\mathbf{1})\right) \\ &\quad\times\exp\left(-\frac{1}{2}\ln\det[\beta\boldsymbol{I}+\boldsymbol{\Sigma}_s\otimes\hat{\boldsymbol{Q}}+\boldsymbol{\Sigma}_n\otimes\hat{\boldsymbol{M}}]\right). \qquad (41)\end{aligned}$$

We now make the replica-symmetric ansatz

$$\begin{aligned}\boldsymbol{Q}=q\boldsymbol{I}+q_0\mathbf{1}\mathbf{1}^\top\ ,\ \boldsymbol{M}=m\boldsymbol{I}+m_0\mathbf{1}\mathbf{1}^\top\ ,\ \boldsymbol{R}=r\boldsymbol{I}+r_0\mathbf{1}\mathbf{1}^\top \\ \hat{\boldsymbol{Q}}=\hat{q}\boldsymbol{I}+\hat{q}_0\mathbf{1}\mathbf{1}^\top\ ,\ \hat{\boldsymbol{M}}=\hat{m}\boldsymbol{I}+\hat{m}_0\mathbf{1}\mathbf{1}^\top\ ,\ \hat{\boldsymbol{R}}=\hat{r}\boldsymbol{I}+\hat{r}_0\mathbf{1}\mathbf{1}^\top.\end{aligned} \tag{42}$$

Next we consider the $n\to 0$ limit under this ansatz. First we re-scale order parameters so that $(q,q_0,r,r_0,m,m_0)\to\beta^{-1}(q,q_0,r,r_0,m,m_0)$ and $(\hat{q},\hat{q}_0,\hat{r},\hat{r}_0,\hat{m},\hat{m}_0)\to\beta(\hat{q},\hat{q}_0,\hat{r},\hat{r}_0,\hat{m},\hat{m}_0)$. After simplifying each term and keeping only the leading order terms in n, we get

$$\langle Z^n\rangle=\int dq dq_0 dm dm_0 dr dr_0 d\hat{q} d\hat{q}_0 d\hat{m} d\hat{m}_0 d\hat{r} d\hat{r}_0\exp\left(\frac{n}{2}S[q,q_0,m,m_0,r,r_0,\hat{q},\hat{q}_0,\hat{m},\hat{m}_0,\hat{r},\hat{r}_0]\right), \tag{43}$$

where the mean field action S has the form

$$\begin{aligned}S&=q\hat{q}+q_0\hat{q}+q\hat{q}_0+m\hat{m}+m_0\hat{m}+m\hat{m}_0+2r\hat{r}+2r_0\hat{r}+2r\hat{r}_0 \\ &-P\left[\ln(\lambda+q+2r+m)+\frac{q_0+2r_0+m_0}{\lambda+q+2r+m}\right]+\frac{1}{N_{\mathcal{H}}}(\hat{r}^2mq+2\hat{r}_0\hat{r}mq+\hat{r}^2mq_0+\hat{r}^2qm_0) \\ &-\beta\hat{q}\boldsymbol{w}^*\boldsymbol{\Sigma}_s\boldsymbol{w}^*+\beta\boldsymbol{w}^*[\hat{q}\boldsymbol{\Sigma}_s][\boldsymbol{I}+\hat{q}\boldsymbol{\Sigma}_s+\hat{m}\boldsymbol{\Sigma}_n]^{-1}[\hat{q}\boldsymbol{\Sigma}_s]\boldsymbol{w}^* \\ &-\ln\det[\boldsymbol{I}+\hat{q}\boldsymbol{\Sigma}_s+\hat{m}\boldsymbol{\Sigma}_n]+\mathrm{Tr}[\boldsymbol{I}+\hat{q}\boldsymbol{\Sigma}_s+\hat{m}\boldsymbol{\Sigma}_n]^{-1}[\hat{q}_0\boldsymbol{\Sigma}_s+\hat{m}_0\boldsymbol{\Sigma}_n] \\ &-J(q+q_0+2r+2r_0+m+m_0).\end{aligned} \tag{44}$$

The relevant saddle point equations are

$$\begin{aligned}
\frac{\partial S}{\partial q_0} &= \hat{q} - \frac{P}{\lambda + q + 2r + m} + \frac{1}{N_{\mathcal{H}}}\hat{r}^2 m - J = 0, \\
\frac{\partial S}{\partial r_0} &= 2\hat{r} - \frac{2P}{\lambda + q + 2r + m} - 2J = 0, \\
\frac{\partial S}{\partial m_0} &= \hat{m} - \frac{\alpha}{\lambda + q + 2r + m} + \frac{1}{N_{\mathcal{H}}}\hat{r}^2 q - J = 0, \\
\frac{\partial S}{\partial \hat{q}_0} &= q - \text{Tr}[\boldsymbol{I} + \hat{q}\boldsymbol{\Sigma}_s + \hat{m}\boldsymbol{\Sigma}_n]^{-1}\boldsymbol{\Sigma}_s = 0, \\
\frac{\partial S}{\partial \hat{r}_0} &= 2r + \frac{1}{N_{\mathcal{H}}}\hat{r} m q = 0, \\
\frac{\partial S}{\partial \hat{m}_0} &= m - \text{Tr}[\boldsymbol{I} + \hat{q}\boldsymbol{\Sigma}_s + \hat{m}\boldsymbol{\Sigma}_n]^{-1}\boldsymbol{\Sigma}_n = 0.
\end{aligned} \tag{45}$$

We can express most of these equations entirely in terms the function $\kappa = \lambda + q + 2r + m$ and the matrix $\boldsymbol{G} = (\boldsymbol{I} + \hat{q}\boldsymbol{\Sigma}_s + \hat{m}\boldsymbol{\Sigma}_n)^{-1}$

$$\hat{q} = P\kappa^{-1} + J - \frac{1}{N_{\mathcal{H}}}[P\kappa^{-1} + J]^2 \text{Tr}\boldsymbol{G}\boldsymbol{\Sigma}_n \tag{46}$$

$$\hat{r} = P\kappa^{-1} + J \tag{47}$$

$$\hat{m} = P\kappa^{-1} + J - \frac{1}{N_{\mathcal{H}}}[P\kappa^{-1} + J]^2 \text{Tr}\boldsymbol{G}\boldsymbol{\Sigma}_s \tag{48}$$

$$\kappa = \lambda + \text{Tr}\boldsymbol{G}\boldsymbol{\Sigma}_+ - \frac{2}{N_{\mathcal{H}}}(\text{Tr}\boldsymbol{G}\boldsymbol{\Sigma}_s)(\text{Tr}\boldsymbol{G}\boldsymbol{\Sigma}_n)[P\kappa^{-1} + J]. \tag{49}$$

These are the full saddle point equations. We can compute the generalization error using

$$\begin{aligned}
E_g &= \frac{\partial}{\partial J}|_{J=0}\boldsymbol{w}^* \left[\hat{q}\boldsymbol{\Sigma}_s - \hat{q}\boldsymbol{\Sigma}_s\boldsymbol{G}\hat{q}\boldsymbol{\Sigma}_s\right]\boldsymbol{w}^* = \frac{\partial}{\partial J}|_{J=0}\boldsymbol{w}^* \left[\hat{q}\boldsymbol{\Sigma}_s\boldsymbol{G}(\boldsymbol{I} + \hat{m}\boldsymbol{\Sigma}_n)\right]\boldsymbol{w}^* \\
&= \boldsymbol{w}^* \left[(\partial_J\hat{q})\boldsymbol{\Sigma}_s\boldsymbol{G}(\boldsymbol{I} + \hat{m}\boldsymbol{\Sigma}_n) - \hat{q}\boldsymbol{\Sigma}_s\boldsymbol{G}[\partial_J\hat{q}\boldsymbol{\Sigma}_s + \partial_J\hat{m}\boldsymbol{\Sigma}_n]\boldsymbol{G}(\boldsymbol{I} + \hat{m}\boldsymbol{\Sigma}_n)\right]\boldsymbol{w}^* \\
&+ \boldsymbol{w}^*[\hat{q}\boldsymbol{\Sigma}_s\boldsymbol{G}\boldsymbol{\Sigma}_n\partial_J\hat{m}]\boldsymbol{w}^* \\
&= \boldsymbol{w}^* \left[(\partial_J\hat{q})\boldsymbol{\Sigma}_s\boldsymbol{G}^2(\boldsymbol{I} + \hat{m}\boldsymbol{\Sigma}_n) + (\hat{m}\partial_J\hat{q} - \hat{q}\partial_J\hat{m})\boldsymbol{\Sigma}_s\boldsymbol{G}\boldsymbol{\Sigma}_n\boldsymbol{G}(\boldsymbol{I} + \hat{m}\boldsymbol{\Sigma}_n)\right]\boldsymbol{w}^* \\
&+ \hat{q}\partial_J\hat{m}\boldsymbol{w}^*\boldsymbol{\Sigma}_s\boldsymbol{G}\boldsymbol{\Sigma}_n\boldsymbol{w}^*.
\end{aligned} \tag{50}$$

D.2 Heavily Overparameterized (Large RKHS Dimension) Approximation

We will now take the $N_{\mathcal{H}} \gg P$ approximation. In this limit, we can disregard the corrections which couple $\hat{r}, m, q$ since they occur at a scale of $\frac{1}{N_{\mathcal{H}}}$. This recovers

$$\hat{q} = \hat{m} = P\kappa^{-1} + J \ , \ \kappa = \lambda + \text{Tr}[\boldsymbol{I} + (P\kappa^{-1} + J)\boldsymbol{\Sigma}_+]^{-1}\boldsymbol{\Sigma}_+ \tag{51}$$

Using our above formula under this assumption, using $\hat{m} = \hat{q}$ we find

$$E_g = (\partial_J\hat{q})\boldsymbol{w}^* \left[\boldsymbol{\Sigma}_s\boldsymbol{G}^2(\boldsymbol{I} + \hat{q}\boldsymbol{\Sigma}_n) + \boldsymbol{\Sigma}_s\boldsymbol{G}\boldsymbol{\Sigma}_n\right]\boldsymbol{w}^*. \tag{52}$$

To compute $\partial_J\hat{q}$, we need to evaluate $\partial_J\kappa(J)$, which can be obtained implicitly

$$\kappa'(J) = P\kappa^{-2}\kappa'(J)\text{Tr}\boldsymbol{G}^2\boldsymbol{\Sigma}_+^2 - \text{Tr}\boldsymbol{G}^2\boldsymbol{\Sigma}_+^2 \tag{53}$$

$$\implies \kappa'(J) = -P^{-1}\kappa^2\frac{\gamma}{1-\gamma} \ , \ \gamma = P\kappa^{-2}\text{Tr}\boldsymbol{G}^2\boldsymbol{\Sigma}_+^2,$$

$$\hat{q}'(J) = -P\kappa^{-2}\kappa'(J) + 1 = \frac{\gamma}{1-\gamma} + 1 = \frac{1}{1-\gamma}. \tag{54}$$

The generalization error under this approximation is thus

$$E_g = \frac{1}{1-\gamma}\boldsymbol{w}^*\boldsymbol{\Sigma}_s\boldsymbol{G}[\boldsymbol{G}(\boldsymbol{I} + \hat{q}\boldsymbol{\Sigma}_n) + \hat{q}\boldsymbol{\Sigma}_n]\boldsymbol{w}^*. \tag{55}$$

This error will plateau to a non-zero asymptote at large P

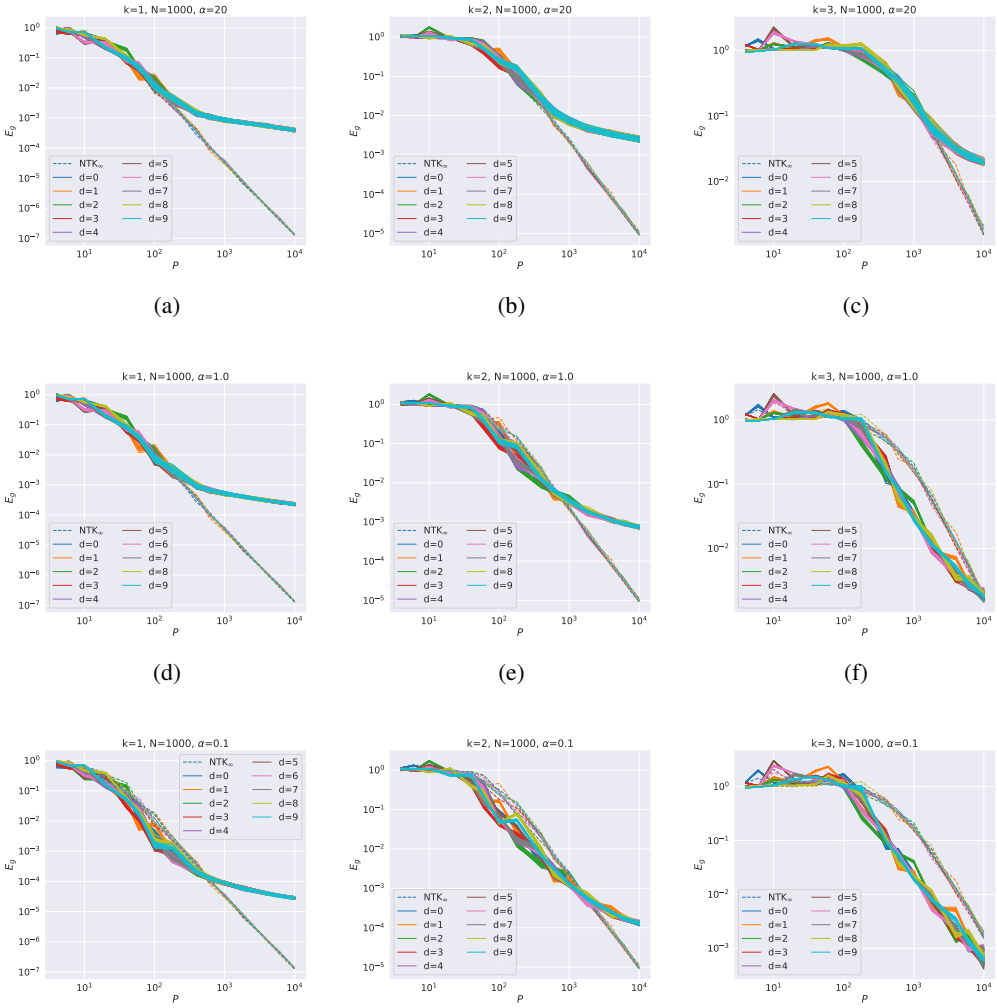
$$\lim_{P\to\infty} E_g = \boldsymbol{w}^* \boldsymbol{\Sigma}_s [\boldsymbol{\Sigma}_s + \boldsymbol{\Sigma}_n]^{-1} \boldsymbol{\Sigma}_n \boldsymbol{w}^*. \tag{56}$$

Suppose for simplicity that $\boldsymbol{\Sigma}_n = \sigma^2 \boldsymbol{I}$. Then, we have

$$E_g = \frac{1}{1-\gamma} \sum_k \frac{\lambda_k w_k^2}{(1 + P\kappa^{-1}(\lambda_k + \sigma^2))} \left[\frac{1 + \sigma^2 P \kappa^{-1}}{(1 + P\kappa^{-1}(\lambda_k + \sigma^2))} + P\kappa^{-1}\sigma^2 \right]. \tag{57}$$

For the purposes of extracting scaling laws, $\gamma \sim 0$ at large P Bordelon et al. (2020); Bahri et al. (2021). This final approximation gives the expression in the main text.

E Additional Plots

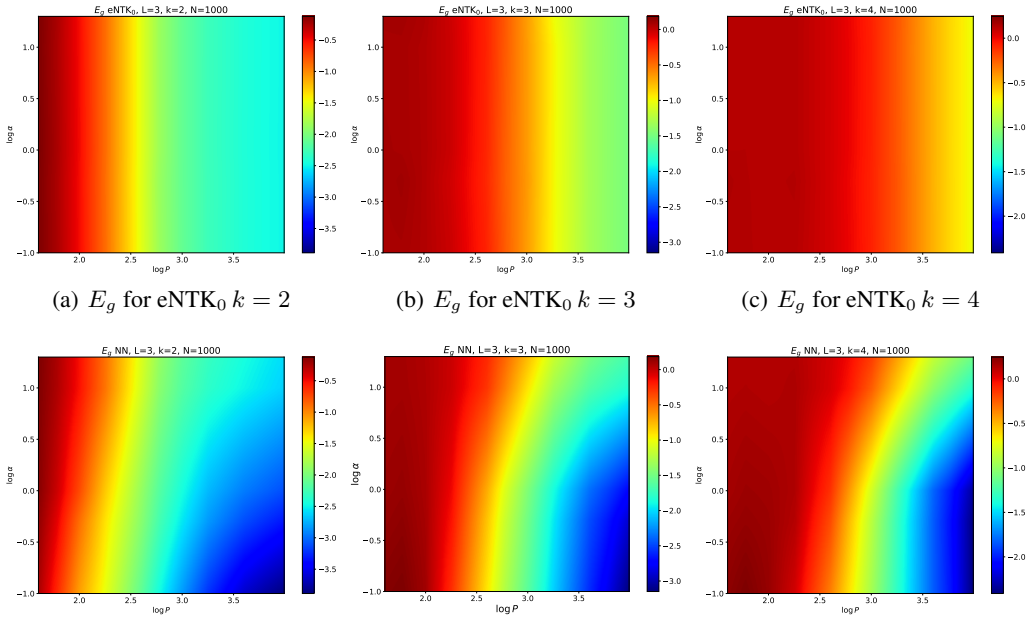


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

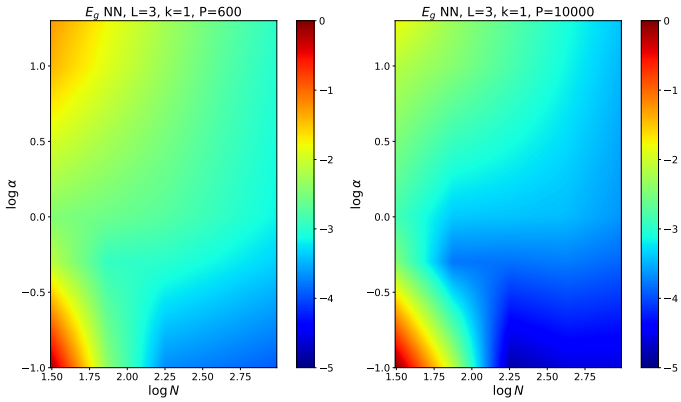
Figure 8: A fine-grained view of the generalization error across different datasets and ensembles. Solid curves are depth 3 neural networks, dashed curves are the infinite width NTK (which only has variance over datasets). Each color is a set of networks trained on the same dataset but different initializations. Different colors correspond to different datasets indexed by $d \in \{0, \ldots, 9\}$.



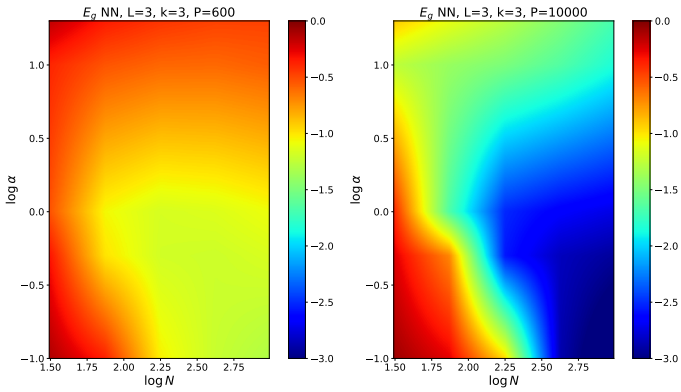
(a) E_g for eNTK$_0$ $k = 2$ (b) E_g for eNTK$_0$ $k = 3$ (c) E_g for eNTK$_0$ $k = 4$

(d) E_g for NN $k = 2$ (e) E_g for NN $k = 3$ (f) E_g for NN $k = 4$

Figure 9: Phase plots of $\log_{10} E_g$ for initial eNTKs (top) and neural networks (bottom). The large α behavior of the neural network generalization matches the generalization of the corresponding eNTK$_0$. As a sanity check, the eNTK$_0$ generalization error is independent of re-scaling the network initialization because of the homogeneity of the ReLU network output.

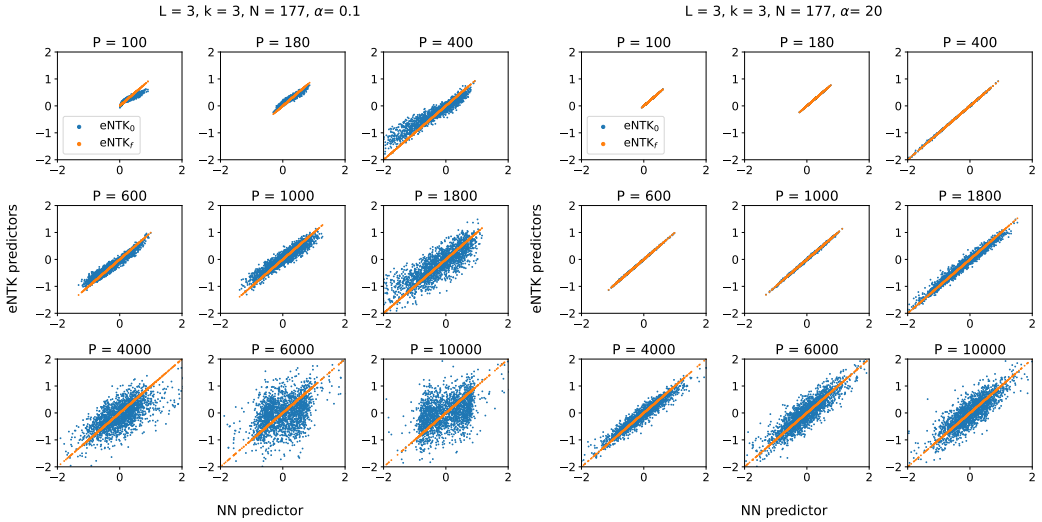


(a) E_g for $k = 1, P = 600$ (b) E_g for $k = 1, P = 10000$

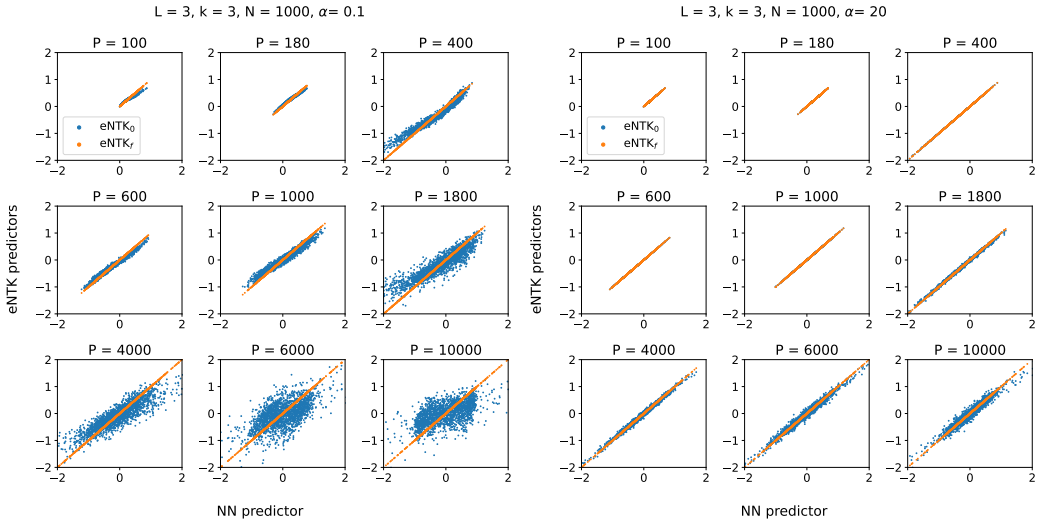


(c) E_g for $k = 3, P = 600$ (d) E_g for $k = 3, P = 10000$

Figure 10: Phase plots of $\log_{10} E_g$ for neural networks in the N-α plane. We plot these at different train set sizes P and different tasks k. The colors are fixed to match across networks trained on the same task.

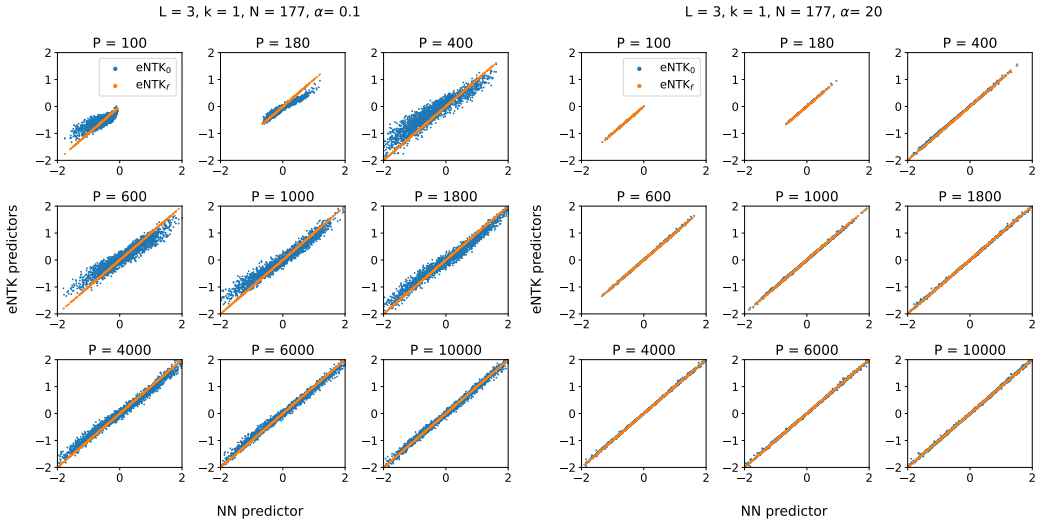


(a) $k = 3, N = 177, \alpha = 0.1$ (b) $k = 3, N = 177, \alpha = 20$



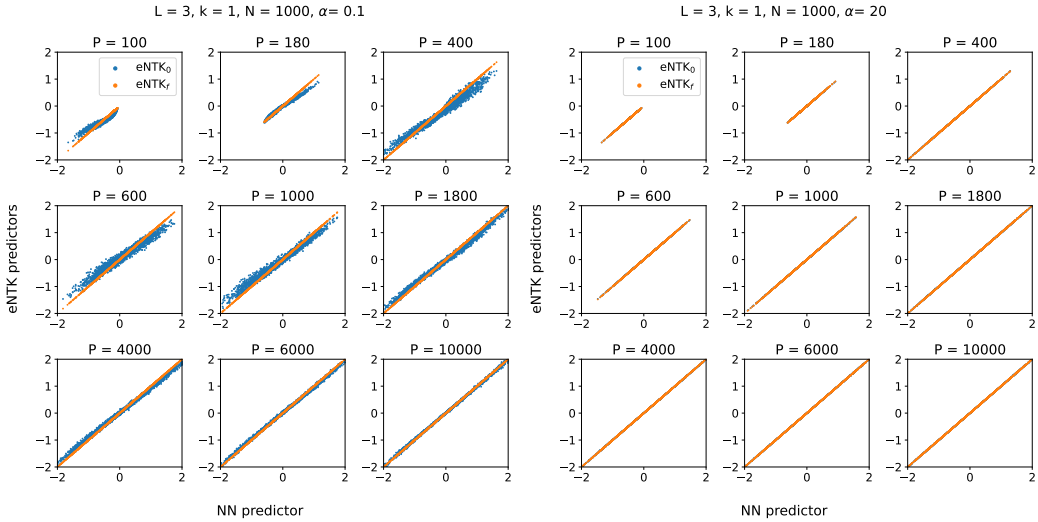
(c) $k = 3, N = 1000, \alpha = 0.1$ (d) $k = 3, N = 1000, \alpha = 20$

Figure 11: Comparison of the neural network predictor (x-axis) to eNTK$_0$, eNTK$_f$ (blue, orange respectively, y-axis) across several training dataset sizes. a) $N = 177, \alpha = 0.1$, b) $N = 177, \alpha = 20$, c) $N = 1000, \alpha = 0.1$ d) $N = 1000, \alpha = 20$. All networks are depth 3. Note how in the rich regime there is a much stronger distinction between the eNTK$_0$ and the neural network. In all regimes, eNTK$_f$ matches the NN output.



(a) $k = 1, N = 177, \alpha = 0.1$

(b) $k = 1, N = 177, \alpha = 20$



(c) $k = 1, N = 1000, \alpha = 0.1$

(d) $k = 1, N = 1000, \alpha = 20$

Figure 12: The same as figure 11 but for fitting a linear $k = 1$ mode. Because the task is simpler, it doesn't require as large of an α to enter the lazy regime.