NEURAL NETWORKS AS KERNEL LEARNERS: THE SILENT ALIGNMENT EFFECT

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks in the lazy training regime converge to kernel machines. Can neural networks in the rich feature learning regime learn a kernel machine with a data-dependent kernel? We demonstrate that this can indeed happen due to a phenomenon we term *silent alignment*, which requires that the NTK of a network evolves in eigenstructure while small in overall scale and before the loss appreciably decreases. We show that such an effect takes place in homogenous neural networks with small initialization and approximately whitened data. We provide an analytical treatment of this effect in the linear network case. In general, we find that the kernel develops a low-rank contribution in the early phase of training, and then evolves in overall scale, yielding a function equivalent to a kernel regression solution with the final network's NTK. The early spectral learning of the kernel depends on both depth and on relative learning rates in each layer. We also demonstrate that non-whitened data can weaken the silent alignment effect.

1 INTRODUCTION

Recent work has established that neural networks can operate in two different regimes depending on the scale of the initial network output and the network's parameterization. In the lazy training regime, the network's parameter gradients do not move very much through training, though the network's output interpolates the training data. In such a regime, when trained with gradient flow on MSE loss, the network converges to a kernel machine with a *data-independent* kernel known as the neural tangent kernel (NTK) (Jacot et al., 2018; Arora et al., 2019; 2020; Lee et al., 2019). The correspondence between networks in the lazy regime and kernel machines has motivated many works exploring how the lazy regime differs from the rich regime of neural network training, where the network learns data-dependent features (Ghorbani et al., 2019; Lee et al., 2020; Yang & Hu, 2020; Ghorbani et al., 2020; Geiger et al., 2021; Paccolat et al., 2021; Liu et al., 2021; Yang, 2021; Woodworth et al., 2020). Some works have proposed to track how the NTK evolves during representation learning in the rich feature learning regime: for example, perturbative corrections (Dyer & Gur-Ari, 2020; Aitken & Gur-Ari, 2020; Andreassen & Dyer, 2020; Zavatone-Veth et al., 2021) and phenomenological models of kernel evolution (Baratin et al., 2021; Huang & Yau, 2020; Shan & Bordelon, 2021) have also been recently invoked to gain insight into feature learning in rich neural networks.

In this present study, we ask whether a neural network trained in the rich feature learning regime, can ever be a kernel machine for a *data-dependent* kernel: i.e. does there exist a kernel function K for which the final neural network function f is $f(\boldsymbol{x}) \approx \sum_{\mu=1}^{P} \alpha^\mu K(\boldsymbol{x}, \boldsymbol{x}^\mu)$ with coefficients α^μ that depend only on the training data? We answer in the affirmative: that under certain special circumstances richly trained networks are accurately approximated as kernel regression solutions with their final, data-dependent NTKs. This finding is consistent with prior empirical observations which found that the outputs of a fully trained neural network in the rich regime were strongly correlated with the predictions of a kernel machine for the final NTK (Geiger et al., 2021; Paccolat et al., 2021). This question was also recently studied by Domingos (2020), who showed that every model trained with gradient descent leads to a kernel model with a path kernel and coefficients α^μ that depend on the test point $\boldsymbol{x}$. This dependence on $\boldsymbol{x}$ makes the construction of Domingos (2020) not a kernel method in the traditional sense that we consider here (see Remark 1 in (Domingos, 2020)).

The phenomenon we study consists of two training phases. In the first phase, the kernel starts off small in overall scale and quickly aligns its eigenvectors toward task-relevant directions. In the second phase, the kernel increases in overall scale, causing the network to learn a kernel regression solution with the final NTK. We call this phenomenon the *silent alignment effect* because the feature learning happens before the loss appreciably decreases.

Concretely, our contributions are the following

1. In section 2, we demonstrate in theory how the silent alignment effect works by considering a simplified model where the kernel evolves while small and then subsequently increases only in scale. We show that if these conditions are met, the final neural network is a kernel machine that uses the final NTK.
2. In section 3, we provide an analysis of the NTK evolution of single hidden layer linear neural networks with scalar target function with small initialization. If the input training data is approximately whitened, the kernel aligns its eigenvectors towards the direction of the optimal linear function early on while the loss does not decrease appreciably. After this the kernel changes in scale only, showing linear networks with small initialization satisfy the requirements for silent alignment discussed in Section 2.
3. In section 4, we extend our analysis to deep linear networks by showing how the time required for alignment scales with initialization versus the time for the loss to decrease appreciably. We present an explicit formula for the final kernel in linear networks of any depth and width when trained on whitened data starting from small initialization.
4. In section 5, we show empirically that the silent alignment phenomenon carries over to nonlinear networks trained with ReLU and Tanh activations on isotropic data, as well as linear and nonlinear networks with multiple output classes. For anisotropic data, we show that the NTK must necessarily change its eigenvectors when the loss is significantly decreasing, destroying the silent alignment phenomenon. In these cases, the final neural network is not a kernel machine that uses the final NTK.

2 The Silent Alignment Effect and Approximate Kernel Solution

Neural networks in the over-parameterized regime can find many interpolators: the precise function that the network converges to is controlled by the time evolution of the NTK. As a concrete example, we will consider learning a scalar target function with mean square error loss through gradient flow. Let $\boldsymbol{x} \in \mathbb{R}^D$ represent an arbitrary input to the network $f(\boldsymbol{x})$ and let $\{\boldsymbol{x}^\mu, y^\mu\}_{\mu=1}^P$ be a supervised learning training set. Under gradient flow the parameters $\boldsymbol{\theta}$ of the neural network will evolve, so the output function is time-dependent and we write this as $f(\boldsymbol{x}, t)$. The evolution for the predictions of the network on a test point can be written in terms of the neural tangent kernel $K(\boldsymbol{x}, \boldsymbol{x}', t) = \frac{\partial f(\boldsymbol{x},t)}{\partial \boldsymbol{\theta}} \cdot \frac{\partial f(\boldsymbol{x}',t)}{\partial \boldsymbol{\theta}}$ as

$$\frac{d}{dt} f(\boldsymbol{x}, t) = \eta \sum_\mu K(\boldsymbol{x}, \boldsymbol{x}^\mu, t)(y^\mu - f(\boldsymbol{x}^\mu, t)), \tag{1}$$

where η is the learning rate. If one had access to the dynamics of $K(\boldsymbol{x}, \boldsymbol{x}^\mu, t)$ throughout all t, one could solve for the final learned function f^* with integrating factors

$$f^*(\boldsymbol{x}) = \sum_{\mu\nu} \int_0^\infty dt\, \boldsymbol{k}_t(\boldsymbol{x})^\mu \exp\left(-\eta \int_0^t \boldsymbol{K}(t')dt'\right)_{\mu\nu} (y^\nu - f_0(\boldsymbol{x}^\nu))\,. \tag{2}$$

Here, $\boldsymbol{k}_t(\boldsymbol{x})^\mu = K(\boldsymbol{x}, \boldsymbol{x}^\mu, t)$, $[\boldsymbol{K}_t]_{\mu,\nu} = K(\boldsymbol{x}^\mu, \boldsymbol{x}^\nu, t)$, and $y^\mu - f_0(\boldsymbol{x}^\mu)$ is the initial error on point $\boldsymbol{x}^\mu$. We see that the final function has contributions throughout the full training interval $t \in (0, \infty)$. The seminal work by Jacot et al. (2018) considers a limit of wide neural networks, where the kernel function $K_t(\boldsymbol{x}, \boldsymbol{x}')$ stays constant throughout training time. In this setting where the kernel is constant and $f_0(\boldsymbol{x}^\mu) \approx 0$, then we obtain a true kernel regression solution $f(\boldsymbol{x}) = \sum_{\mu,\nu} \boldsymbol{k}(\boldsymbol{x})^\mu \boldsymbol{K}^{-1}_{\mu\nu} y^\nu$ for a kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ which does not depend on the training set.

Much less is known about what happens in the rich, feature learning regime of neural networks, where the kernel evolves significantly during time in a data-dependent manner. In this paper, we

consider a setting where the initial kernel is small in scale, aligns its eigenfunctions and relative eigenvalues early on during gradient descent, and then increases only in scale monotonically. As a concrete phenomenological model, consider starting from a small initial kernel and initial function $K(\boldsymbol{x}, \boldsymbol{x}'), f(\boldsymbol{x}) \sim O(\epsilon)$ and evolving the kernel in scale in some constant direction

$$K(\boldsymbol{x}, \boldsymbol{x}', t) = \begin{cases} \epsilon \tilde{K}(\boldsymbol{x}, \boldsymbol{x}', t) & t \leq \tau \\ g(t) K_\infty(\boldsymbol{x}, \boldsymbol{x}') & t > \tau \end{cases}, \tag{3}$$

where $\tilde{K}(\boldsymbol{x}, \boldsymbol{x}', t)$ evolves from an initial kernel at time $t = 0$ to $K_\infty(\boldsymbol{x}, \boldsymbol{x}')$ by $t = \tau$ and $g(t)$ increases monotonically from ϵ to 1. In this model, one also obtains a kernel regression solution in the limit where $\epsilon \to 0$ with the final, rather than the initial kernel: $f(\boldsymbol{x}) = \boldsymbol{k}_\infty(\boldsymbol{x}) \cdot \boldsymbol{K}_\infty^{-1} \boldsymbol{y} + O(\epsilon)$. We provide a proof of this in the Appendix A.

The assumption that the kernel evolves early on in gradient descent before increasing only in scale may seem overly strict as a model of kernel evolution. However, we analytically show in sections 3 and 4 analytically that this can happen in deep linear networks initialized with small weights, and consequently that the final learned function is a kernel regression for the final NTK. Moreover, we show that for a linear network trained on whitened data, the final NTK is universal across initializations provided that the initialization scale is sufficiently small. A similar conclusion was recently obtained by Stöger & Soltanolkotabi (2021) in the setting of low-rank matrix recovery. The second phase after alignment, where the kernel's eigenspectrum grows, was studied in linear networks in (Jacot et al., 2021), where it is referred to as the saddle-to-saddle regime. Unlike prior work (Dyer & Gur-Ari, 2020; Aitken & Gur-Ari, 2020; Andreassen & Dyer, 2020), we stress that our results do not rely on perturbative expansions in network width, but rely rather on a separation of training timescales that follows from small initialization. Also unlike prior work Saxe et al. (2014), our analytical solutions for the evolution of the kernel do not depend on choosing a specific set of initial conditions, but rather follow only from assumptions of small initialization and whitened data.

Further, we show empirically that our results carry over to nonlinear networks with homogenous activation functions $\phi(ax) = a\phi(x)$ such a ReLU under certain conditions on the data. For example, see Figure 1, where we show the silent alignment effect on ReLU networks with whitened MNIST and CIFAR-10 images. We define alignment as the overlap between the kernel and the target function $\frac{\boldsymbol{y}^\top \boldsymbol{K} \boldsymbol{y}}{\|\boldsymbol{K}\|_F |\boldsymbol{y}|^2}$, where $\boldsymbol{y} \in \mathbb{R}^P$ is a vector of the target values, quantifying the projection of the labels onto the kernel, as discussed in (Cortes et al., 2012). This quantity increases early in training but quickly stabilizes before the loss decreases. Our results suggest that the final NTK may be useful for analyzing generalization and transfer as we discuss for the linear case in Appendix E.

3 Kernel Evolution in 2 Layer Linear Networks

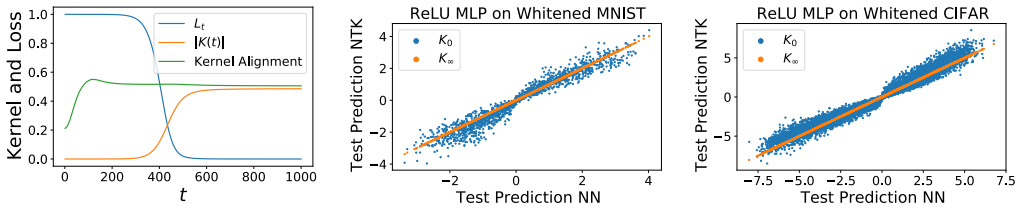
We will first study shallow linear networks trained with small initialization before providing analysis for deeper networks in section 4. We will focus our discussion in this section on the scalar output case but we will provide similar analysis in the multiple output channel case in a subsequent section. We will assume the P data points $\boldsymbol{x}^\mu \in \mathbb{R}^D, \mu = 1, \ldots, P$ have correlation matrix $\boldsymbol{\Sigma} = \frac{1}{P} \sum_{\mu=1}^P \boldsymbol{x}^\mu \boldsymbol{x}^{\mu\top}$. Further, we will assume that the target values are generated by a linear teacher function $y^\mu = s \boldsymbol{\beta}_T \cdot \boldsymbol{x}^\mu$ for a unit vector $\boldsymbol{\beta}_T$. The scalar s merely quantifies the size of the supervised learning signal: the variance of $|\boldsymbol{y}|^2 = s^2 \boldsymbol{\beta}_T^\top \boldsymbol{\Sigma} \boldsymbol{\beta}_T$. We define the two-layer linear neural network with N hidden units as $f(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{W} \boldsymbol{x}$. Concretely, we initialize the weights with standard parameterization $a_i \sim \mathcal{N}(0, \sigma^2/N), W_{ij} \sim \mathcal{N}(0, \sigma^2/D)$. Understanding the role of σ in the dynamics will be crucial to our study. We analyze gradient flow dynamics on MSE cost $L = \frac{1}{2P} \sum_\mu \left(f(\boldsymbol{x}^\mu) - y^\mu\right)^2$. Under gradient flow with learning rate $\eta = 1$, the weight matrices in each layer evolve as

$$\frac{d}{dt} \boldsymbol{a} = -\frac{\partial L}{\partial \boldsymbol{a}} = \boldsymbol{W} \boldsymbol{\Sigma} \left(s \boldsymbol{\beta}_T - \boldsymbol{W}^\top \boldsymbol{a}\right), \quad \frac{d}{dt} \boldsymbol{W} = -\frac{\partial L}{\partial \boldsymbol{W}} = -\boldsymbol{a} \left(s \boldsymbol{\beta}_T - \boldsymbol{W}^\top \boldsymbol{a}\right)^\top \boldsymbol{\Sigma}. \tag{4}$$

The NTK takes the following form throughout training.

$$K(\boldsymbol{x}, \boldsymbol{x}'; t) = \boldsymbol{x}^\top \boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{x}' + |\boldsymbol{a}|^2 \boldsymbol{x}^\top \boldsymbol{x}'. \tag{5}$$

Note that while the second term, a simple isotropic linear kernel, does not reflect the nature of the learning task, the first term $\boldsymbol{x}^\top \boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{x}'$ can evolve to yield an an anisotropic kernel that has learned a representation from the data.



(a) Whitened Data Dynamics (b) Prediction MNIST (c) Prediction CIFAR-10

Figure 1: A demonstration of the Silent Alignment effect: depth-2 MLPs trained on whitened data and small initialization are accurately modeled as kernel regression with the final NTK $\boldsymbol{K}_\infty$. (a) We trained a 2-layer ReLU MLP on $P = 1000$ MNIST images of handwritten zeros and ones which were whitened. Early in training, around $t \approx 50$, the NTK eigenvectors and relative eigenvalues align to the target function and stay fixed (green). The kernel's overall scale (orange) and the loss (blue) begin to move at the same time around $t = 300$. (b) We compare the predictions of the NTK regressors and the trained network on MNIST test points. As a consequence of the separation in timescales between alignment and loss evolution, the final learned function is well described as a kernel regression solution with the final kernel. Different kernels can converge to different interpolators, as evidenced by the poor match between the NN and the initial NTK $\boldsymbol{K}_0$, showing that the initial NTK is not a good model of the network's predictions. (c) The same experiment on $P = 1000$ whitened CIFAR-10 images from the first two classes. Here we use MSE loss with 100 neurons in the hidden layer with gaussian initialization having $\sigma = 0.1$.

3.1 Phases of Training in Two Layer Linear Network

In the following discussion, we will show that there are essentially two phases of training when training a two-layer linear network from small initialization on whitened-input data.

- Phase I: An alignment phase which occurs for $t \sim \frac{1}{s}$. In this phase the weights align to their low rank structure and the kernel picks up a rank-one term of the form $\boldsymbol{x}^\top \boldsymbol{\beta}\boldsymbol{\beta}^\top \boldsymbol{x}'$. In this setting the overall scale of the weights and kernel, evolves slowly since the network is initialized close to the saddle point at $\boldsymbol{W}, \boldsymbol{a} = \boldsymbol{0}$.
- Phase II: A data fitting phase which begins around $t \sim \frac{1}{s}\log(s\sigma^{-2})$. In this phase, the system escapes the initial saddle point and loss decreases to zero. In this setting both the kernel's overall scale and the scale of the function $f(\boldsymbol{x}, t)$ increase substantially.

If Phase I and Phase II are well separated in time, which can be guaranteed by making σ small, then the final function solves a kernel interpolation problem for the NTK which is only sensitive to the geometry of gradients in the final basin of attraction. In fact, in the linear case, the kernel at every point along the gradient descent trajectory would give the final solution as we show in Appendix F. A visual summary of these phases is provided in Figure 2.

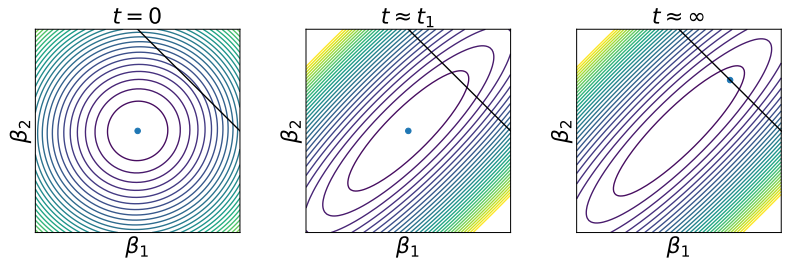
3.1.1 Phase I: Early Alignment for Small Initialization

In this section we will show how the kernel aligns to the correct eigenspace early in training. We approximate the dynamics by recognizing that the network output is small early in training due to the small initialization. We focus on the whitened setting, where the data matrix $\boldsymbol{X}$ has all of its nonzero singular values equal. We let $\boldsymbol{\beta}$ represent the normalized component of $\boldsymbol{\beta}_T$ in the span of the training data $\{\boldsymbol{x}^\mu\}$. We will discuss general $\boldsymbol{\Sigma}$ in section 3.2. We make the following approximation early in training due to small initialization. Early on, the dynamics are approximately

$$\frac{d}{dt}\boldsymbol{a} \sim s\boldsymbol{W}\boldsymbol{\beta} + O(\sigma^3)\,, \quad \frac{d}{dt}\boldsymbol{W} \sim s\boldsymbol{a}\boldsymbol{\beta}^\top + O(\sigma^3). \tag{6}$$

Truncating terms order σ^3 and higher, we can solve for the kernel's dynamics early on in training

$$K(\boldsymbol{x}, \boldsymbol{x}'; t) \sim q_0 \cosh(2\eta s t)\, \boldsymbol{x}^\top \left[\boldsymbol{\beta}\boldsymbol{\beta}^\top + \boldsymbol{I}\right] \boldsymbol{x}' + K_0(\boldsymbol{x}, \boldsymbol{x}'), \tag{7}$$



(a) Initialization (b) Phase 1 (c) Phase 2

Figure 2: The evolution of the kernel's eigenfunctions happens during the early alignment phase for $t_1 \approx \frac{1}{s}$, but significant evolution in the network predictions happens for $t > t_2 = \frac{1}{2}\log(s\sigma^{-2})$. (a) Contour plot of kernel's norm for linear functions $f(\boldsymbol{x}) = \boldsymbol{\beta} \cdot \boldsymbol{x}$. The black line represents the space of weights which interpolate the training set, ie $\boldsymbol{X}^\top \boldsymbol{\beta} = \boldsymbol{y}$. At initialization, the kernel is isotropic, resulting in spherically symmetric level sets of RKHS norm. The network function is represented as a blue dot. (b) During Phase I, the kernel's eigenfunctions have evolved, enhancing power in the direction of the min-norm interpolator, but the network function has not moved far from the origin. (c) In Phase II, the network function $\boldsymbol{W}^\top \boldsymbol{a}$ moves from the origin to the final solution.

where q_0 is an initialization dependent quantity, see Appendix B.2. We see that the kernel picks up a rank one-correction $\boldsymbol{\beta}\boldsymbol{\beta}^\top$ which points in the direction of the task vector $\boldsymbol{\beta}$, indicating that the kernel evolves in a direction sensitive to the target function $y = s\boldsymbol{\beta}_T \cdot \boldsymbol{x}$. This new term grows exponentially during the early stages of training, and overwhelms the original kernel K_0 with timescale $1/s$. Though the neural network achieved low loss in this phase, the alignment of the kernel and learned representation has consequences for the transfer ability of the network on correlated tasks as we show in Appendix E.

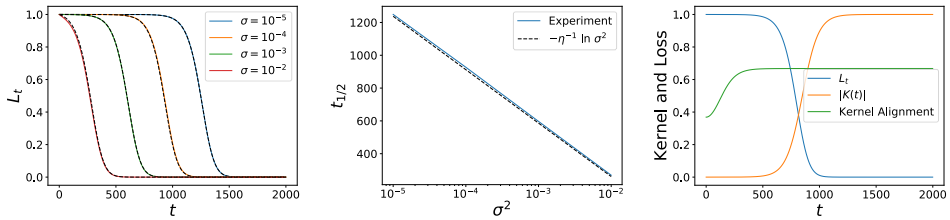
3.1.2 Phase II: Spectral Learning

We now assume that the weights have approached their low rank structure, as predicted from the previous analysis of phase I dynamics and study the subsequent NTK evolution. We will show that, under the assumption of whitening, the kernel only evolves in overall scale.

First, following Arora et al. (2018); Du et al. (2018), we note the following conservation law $\frac{d}{dt}\left[\boldsymbol{a}(t)\boldsymbol{a}(t)^\top - \boldsymbol{W}(t)\boldsymbol{W}(t)^\top\right] = 0$ which holds for all time. If we assume small initialization, the approximation that the weights are balanced: $\boldsymbol{a}\boldsymbol{a}^\top \approx \boldsymbol{W}\boldsymbol{W}^\top$, becomes more and more accurate with error on the order of $O(\sigma^2)$. This condition is surprisingly informative, since it indicates that $\boldsymbol{W}$ is approximately rank-one. From the analysis of the alignment phase, we also have that $\boldsymbol{W}^\top\boldsymbol{W} \propto \boldsymbol{\beta}\boldsymbol{\beta}^\top$. These two observations uniquely determine the rank one structure of $\boldsymbol{W}$ to be $\boldsymbol{a}\boldsymbol{\beta}^\top$. Thus, from equation 5 it follows that in Phase II, the kernel evolution takes the form

$$K(\boldsymbol{x}, \boldsymbol{x}'; t) = u(t)^2 \boldsymbol{x}^\top \left[\boldsymbol{\beta}\boldsymbol{\beta}^\top + \boldsymbol{I}\right] \boldsymbol{x}' + O(\sigma^2) \tag{8}$$

where $u(t)^2 = |\boldsymbol{a}|^2$. This demonstrates that the kernel only changes in overall scale during phase two. Once the weights are aligned with this scheme, we can get an expression for the evolution of $u(t)^2$ analytically $u(t)^2 = \frac{se^{2st}}{e^{2st}-1+s/u_0^2}$ (Saxe et al., 2014). This is a sigmoidal curve which starts at u_0^2 and approaches s. The transition time where active learning begins occurs when $e^{st} \approx s/u_0^2 \implies t \approx s^{-1}\log(s/\sigma^2)$. This analysis demonstrates that the kernel only evolves in scale during this second phase in training from the small initial value $u_0^2 \sim O(\sigma^2)$ to its asymptote. This is equivalent to the assumptions discussed in Section 2, with $g(t) = u(t)^2$, showing that the final solution is well approximated by kernel regression with the final NTK. We stress that the timescale for the first phase $t_1 \sim 1/s$, where eigenvectors evolve, is independent of the scale of the initialization σ^2, whereas the second phase occurs around $t_2 \approx t_1 \log(s/\sigma^2)$. This separation of timescales $t_1 \ll t_2$ for small initialization guarantees the silent alignment effect. We illustrate these learning curves and the time required to escape the saddle point against σ^2, the initialization variance in Figure 3. In this setting, the trained network function is a kernel machine that uses the final NTK



(a) Loss Dynamics (b) Time to $L_t = 0.5$ (c) Alignment $\sigma = 10^{-4}$.

Figure 3: Initialization scale controls the time spent in phase I, where the network escapes the saddle point near $\boldsymbol{W}, \boldsymbol{a} = \boldsymbol{0}$ and the kernel aligns to the task. (a) The loss curves for two-layer linear networks with small initialization follow sigmoidal trajectories as in Saxe et al. (2014) which transition from their maximum to minimum at a time which decreases with initialization scale. Theory is shown in black dashed lines. (b) Verification of the Phase I time $t_{1/2}$, measured as the time for the loss to reach one half its original value. This scales logarithmically with σ^2. (c) The alignment of the kernel eigenfunctions happens before the loss appreciably decreases for $\sigma = 10^{-4}$, evidenced by the kernel alignment curve.

3.2 Unwhitened data

In the case where the data is unwhitened, the right singular vector of $\boldsymbol{W}$ aligns with $\boldsymbol{\Sigma\beta}$ early in training, as we show in Appendix B.3. This happens since, early on, the dynamics for the first layer are $\frac{d}{dt}\boldsymbol{W} \sim \boldsymbol{a}(t)\boldsymbol{\beta}^\top\boldsymbol{\Sigma}$. Thus the early time kernel will have a rank-one spike in the $\boldsymbol{\Sigma\beta}$ direction. However, this configuration is not stable as the network outputs grow. In fact, at late time $\boldsymbol{W}$ must realign to converge to $\boldsymbol{W} \propto \boldsymbol{a\beta}^\top$ since the network function converges to the optimum and $f = \boldsymbol{a}^\top\boldsymbol{W}\boldsymbol{x} = s\boldsymbol{\beta}\cdot\boldsymbol{x}$. Thus, the final kernel will always look like $K_\infty(\boldsymbol{x}, \boldsymbol{x}') = s\boldsymbol{x}^\top\left[A\boldsymbol{\beta\beta}^\top + B\boldsymbol{I}\right]\boldsymbol{x}'$. However, since the realignment of $\boldsymbol{W}$'s singular vectors happens *during the phase II spectral learning*, the kernel is not constant up to overall scale, violating the conditions for silent alignment. This suggests that anisotropy can prevent the final predictor from being a NTK regressor.

4 Extension to Deep Linear Networks

We next consider scalar target functions approximated by deep linear neural networks and show that many of the insights from the two layer network carry over. The neural network function $f : \mathbb{R}^D \to \mathbb{R}$ takes the form $f(\boldsymbol{x}) = \boldsymbol{w}^{L\top}\boldsymbol{W}^{L-1}...\boldsymbol{W}^1\boldsymbol{x}$. The gradient flow dynamics under mean squared error (MSE) loss become

$$\frac{d}{dt}\boldsymbol{W}^\ell = -\eta_\ell\frac{\partial L}{\partial \boldsymbol{W}^\ell} = \eta_\ell\left(\prod_{\ell'>\ell}\boldsymbol{W}^{\ell'}\right)^\top(s\boldsymbol{\beta} - \tilde{\boldsymbol{w}})^\top\boldsymbol{\Sigma}\left(\prod_{\ell'<\ell}\boldsymbol{W}^{\ell'}\right)^\top, \tag{9}$$

where $\tilde{\boldsymbol{w}} = \boldsymbol{W}^{1\top}\boldsymbol{W}^{2\top}...\boldsymbol{w}^L \in \mathbb{R}^D$ is shorthand for the effective one-layer linear network weights. Here, we have included the possibility of using different learning rates η_ℓ for different layers. Inspired by observations made in prior works (Arora et al., 2018; Du et al., 2018), we again note that the following set of conservation laws hold during the dynamics of gradient descent $\frac{d}{dt}\left[\frac{1}{\eta_\ell}\boldsymbol{W}^\ell\boldsymbol{W}^{\ell\top} - \frac{1}{\eta_{\ell+1}}\boldsymbol{W}^{\ell+1\top}\boldsymbol{W}^{\ell+1}\right] = 0$. This condition indicates a balance in the size of weight updates in adjacent layers and simplifies the analysis of linear networks. We note, however, that the scale of the weights in each layer ℓ scales with the square root of the learning rate for that layer $\sqrt{\eta_\ell}$. This is not specific to MSE loss, but will also hold for any loss function, see Appendix C. We will use this condition to characterize the NTK's evolution.

4.1 NTK UNDER SMALL INITIALIZATION

We now consider the effects of small initialization. When the initial weight variance $\text{Var}(W_{ij}^{\ell}) \sim \sigma^2/N_{\ell-1}$ is sufficiently small, $\frac{1}{\eta_\ell}\boldsymbol{W}^{\ell}\boldsymbol{W}^{\ell\top} - \frac{1}{\eta_{\ell+1}}\boldsymbol{W}^{\ell+1\top}\boldsymbol{W}^{\ell+1} = O(\sigma^2) \approx 0$ at initialization. The conservation law above implies that these matrices remain approximately equal throughout training. Performing an SVD on each matrix and inductively using the above formula from the last layer to the first, we find that all matrices will be approximately rank-one $\boldsymbol{w}^L = u(t)\boldsymbol{r}_L(t)$, $\boldsymbol{W}^{\ell} = \sqrt{\frac{\eta_\ell}{\eta_L}}u(t)\boldsymbol{r}_{\ell+1}(t)\boldsymbol{r}_\ell(t)^\top$, where $\boldsymbol{r}_\ell(t)$ are unit vectors. Using only this balancing condition, we find that the NTK's dynamics look like

$$K(\boldsymbol{x},\boldsymbol{x}',t) = u(t)^{2(L-1)}\boldsymbol{x}^\top\left[A\boldsymbol{r}_1(t)\boldsymbol{r}_1(t)^\top + B\boldsymbol{I}\right]\boldsymbol{x}' \ , \ A = \sum_{\ell=2}^{L}\prod_{\ell'\neq\ell}\frac{\eta_{\ell'}}{\eta_L} \ , \ B = \prod_{\ell\neq 1}\frac{\eta_\ell}{\eta_L}. \quad (10)$$

We derive this formula in the Appendix D. We observe that the NTK consists of a rank-1 correction to the isotropic linear kernel $\boldsymbol{x}\cdot\boldsymbol{x}'$ with the rank-one spike pointing along the $\boldsymbol{r}_1(t)$ direction. This is true dynamically throughout training. We note that the strength of feature learning A/B, is enhanced with depth L and with an increase in the ratio of η_1/η_L, the ratio of the learning rate for the first and last layer. This is qualitatively similar to the finding in (Yang & Hu, 2020) which shows that to have stable maximal feature learning in an infinite neural network, the first layer should have enhanced effective learning rate compared to the NTK parameterization. We will now discuss the difference of the dynamics in the whitened and unwhitened input data case.

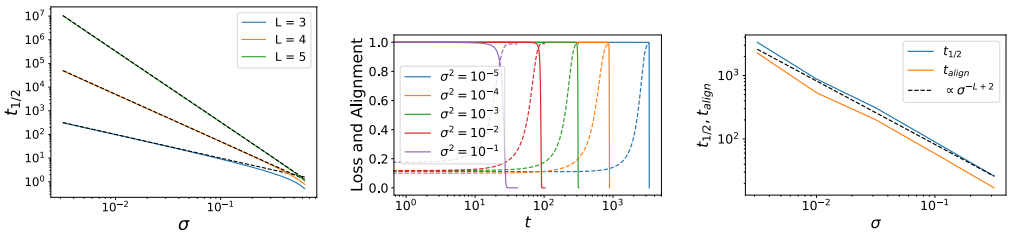
4.1.1 WHITENED DATA VS ANISOTROPIC DATA

In the case where the input data is whitened, the unit vector $\boldsymbol{r}_1(t)$ quickly aligns to $\boldsymbol{\beta}$ since the first layer weight matrix evolves in the rank-one direction $\frac{d}{dt}\boldsymbol{W}^1 = \boldsymbol{v}(t)\boldsymbol{\beta}^\top$ throughout training for a time dependent vector function $\boldsymbol{v}(t)$. As a consequence, early in training the top eigenvector of the NTK aligns to $\boldsymbol{\beta}$. This is merely a consequence of gradient descent. Since the $\boldsymbol{r}_1$ quickly aligns to $\boldsymbol{\beta}$ due to $\boldsymbol{W}^1$ growing only along the $\boldsymbol{\beta}$ direction, then the global scalar function $c(t) = u(t)^L$ satisfies the dynamics $\dot{c}(t) = c(t)^{2-2/L}\left[s - c(t)\right]$ in the whitened data case, which is consistent with the dynamics obtained when starting from the orthogonal initialization scheme of Saxe et al. (2014). We show in the Appendix D.1 that spectral learning occurs over a timescale on the order of $t_{1/2} \approx \frac{L}{s(L-2)}\sigma^{-L+2}$. We discuss this scaling in detail in Figure 4, showing that although the timescale of alignment shares the same scaling with σ for $L > 2$, empirically alignment in deep networks occurs faster than spectral learning. In this setting, the trained network function is again a kernel machine that uses the final NTK.

In the case where the data is unwhitened, the $\boldsymbol{r}_1(t)$ vector aligns with $\boldsymbol{\Sigma}\boldsymbol{\beta}$ early in training. This happens since, early on, the dynamics for the first layer are $\frac{d}{dt}\boldsymbol{W}^1 \sim \boldsymbol{v}(t)\boldsymbol{\beta}^\top\boldsymbol{\Sigma}$ for time dependent vector $\boldsymbol{v}(t)$. However, for the same reasons we discussed in section 3.2 the kernel must realign at late times, violating the conditions for silent alignment.

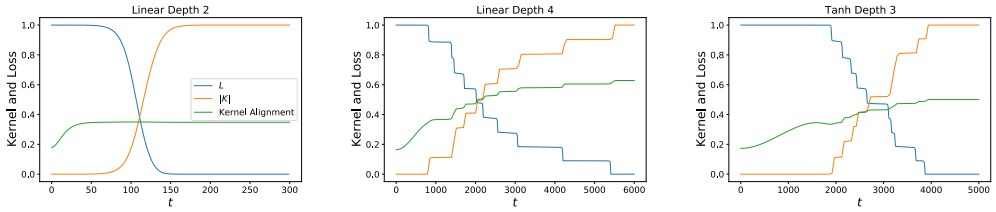
4.2 MULTIPLE OUTPUT CHANNELS

In this subsection, we discuss the case where the network has multiple C output channels. Each network output, we denote as $f_c(\boldsymbol{x}')$ resulting in C^2 kernel sub-blocks $K_{c,c'}(\boldsymbol{x},\boldsymbol{x}') = \nabla f_c(\boldsymbol{x}) \cdot \nabla f_{c'}(\boldsymbol{x})$. In this context, the balanced condition $\boldsymbol{W}^{\ell}\boldsymbol{W}^{\ell\top} \approx \boldsymbol{W}^{\ell+1\top}\boldsymbol{W}^{\ell+1}$ implies that each of the weight matrices is rank-C, implying a rank-C kernel. We give an explicit formula for this kernel in Appendix G. For concreteness, consider whitened input data $\boldsymbol{\Sigma} = \boldsymbol{I}$ and a teacher with weights $\boldsymbol{\beta} \in \mathbb{R}^{C\times D}$. The singular value decomposition of the teacher weights $\boldsymbol{\beta} = \sum_\alpha s_\alpha \boldsymbol{z}_\alpha \boldsymbol{v}_\alpha^\top$ determines the evolution of each mode (Saxe et al., 2014). Each singular mode begins to be learned at $t_\alpha = \frac{1}{s_\alpha}\log\left(s_\alpha u_0^{-2}\right)$. To guarantee silent alignment, we need all of the Phase I time constants to be smaller than all of the Phase II time constants. This is equivalent to the condition $\frac{1}{s_{min}} \ll \frac{1}{s_{max}}\log\left(s_{max}u_0^{-2}\right)$ so that the kernel alignment timescales are well separated from the timescales of spectral learning. We demonstrate that alignment precedes learning in Figure 5.



(a) ODE Time to Learn (b) $L = 3$ Dynamics (c) Time To Learn $L = 3$

Figure 4: Time scales are not strongly separated for deep networks ($L \geq 3$), but alignment is consistently achieved reached before loss decreases. (a) Time to half loss scales in a power law with σ for networks with $L \geq 3$: $t_{1/2} \sim \frac{L}{(L-2)}\sigma^{-L+2}$ (black dashed) is compared with numerically integrating the dynamics $\dot{c}(t) = c^{2-2/L}(s-c)$ (solid). The power law scaling of $t_{1/2}$ with σ is qualitatively different than what happens for $L = 2$, where we identified logarithmic scaling $t_{1/2} \sim \log(\sigma^{-2})$. (b) Linear networks with $D = 30$ inputs and $N = 50$ hidden units trained on synthetic whitened data with $|\boldsymbol{\beta}| = 1$ We show for a $L = 3$ linear network the cosine similarity of $\boldsymbol{W}^{1\top}\boldsymbol{W}^1$ with $\boldsymbol{\beta}\boldsymbol{\beta}^\top$ (dashed) and the loss (solid) for different initialization scales. (c) The time to get to $1/2$ the initial loss and the time for the cosine similarity of $\boldsymbol{W}^{1\top}\boldsymbol{W}^1$ with $\boldsymbol{\beta}\boldsymbol{\beta}^\top$ to reach $1/2$ both scale as σ^{-L+2}, however one can see that alignment occurs before half loss is achieved.



(a) Linear depth 2 (b) Linear depth 4 (c) Tanh depth 3

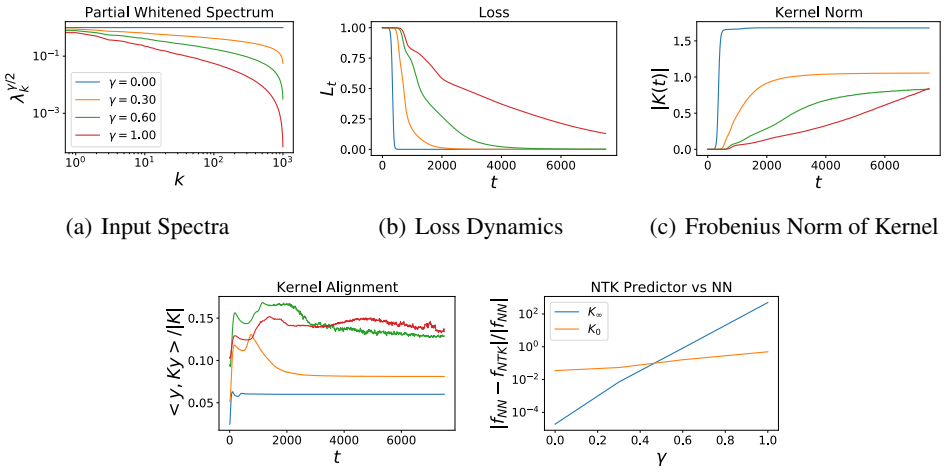
Figure 5: Demonstration of a separation between alignment and spectral learning phases across networks trained on multi-class data. Here we train on whitened MNIST at $\sigma = \{10^{-2}, 10^{-1}, 10^{-2}\}$ respectively. (a) Depth two multi-class dynamics are very similar to single class. (b) For deeper multi-class networks, each singular values is learned far apart in time, and alignment does not as clearly preceed loss decay. (c) Similar dynamics are obtained for tanh networks.

5 Silent Alignment on Real Data and ReLU Nets

In this section, we empirically demonstrate that many of the phenomena described in the previous section carry over to the nonlinear homogenous networks with small initialization provided that the data is not highly anisotropic. A similar separation in timescales is expected in the nonlinear L-homogenous case since, early in training, the kernel evolves more quickly than the network predictions. This argument is based on a phenomenon discussed by Chizat et al. (2019). Consider an initial scaling of the parameters by σ. We find that the relative change in the loss compared to the relative change in the features has the form

$$\frac{|\frac{d}{dt}\nabla f|}{|\nabla f|}\frac{\mathcal{L}}{|\frac{d}{dt}\mathcal{L}|} \approx \frac{|\nabla^2 f|}{\mathcal{L}|\nabla f|^2} = O(\sigma^{-L}) \tag{11}$$

which becomes very large for small initialization σ as we show in Appendix H. This indicates, that from small initialization, the parameter gradients and NTK evolve much more quickly than the loss. This is a necessary, but not sufficient condition for the silent alignment effect. To guarantee the silent alignment, the gradients must be finished evolving except for overall scale by the time the loss appreciably decreases. However, we showed that for whitened data that nonlinear ReLU networks do in fact enjoy the separation of timescales necessary for the silent alignment effect in Figure 1. We now explore how anisotropic data can interfere with silent alignment. We consider



(a) Input Spectra (b) Loss Dynamics (c) Frobenius Norm of Kernel

(d) Phase I alignment (e) Predictor Comparison

Figure 6: Anisotropy in the data introduces multiple timescales which can interfere with the silent alignment effect in a ReLU network. (a) We consider the partial whitening transformation on the 1000 CIFAR-10 images $\lambda_k \to \lambda_k^\gamma$ for $\gamma \in (0, 1)$ for covariance eigenvalues $\boldsymbol{\Sigma}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$. (b) The loss dynamics for unwhitened data has a multitude of timescales rather than a single sigmoidal learning curve. As a consequence, kernel alignment does not happen all at once before the loss decreases and the final solution is not a kernel machine with the final NTK. (c) The kernel continues growing in norm over a much wider range of times. Anisotropic data gives a slower evolution in the kernel's norm. (d) The kernel alignment very rapidly approaches an asymptote for whitened data but exhibits nontrivial dynamics for the anisotropic data. (e) The final NTK predictor gives the same predictions as the neural network only when $\gamma \approx 0$.

the partial whitening transformation: let the singular value decomposition of the data matrix be $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ and construct a new partially whitened dataset $\boldsymbol{X}_\gamma = \boldsymbol{U}\boldsymbol{S}^\gamma\boldsymbol{V}^\top$, where $\gamma \in (0, 1)$. As $\gamma \to 0$ the dataset becomes closer to perfectly whitened. We compute loss and kernel aligment for depth-2 ReLU MLPs on a subset of CIFAR-10 and show results in Figure 6. As $\gamma \to 0$ the agreement between the final NTK regressor and the learned neural network function becomes much closer, since the kernel alignment curve is stable after a smaller number of training steps. As the data becomes more anisotropic, the kernel's dynamics become less trivial at later time: rather than evolving only in scale, the alignment with the target function varies in a non-trivial way while the loss is decreasing. As a consequence, the NN function is not a kernel machine with the final NTK.

6 Conclusion

We provided an example of a case where neural networks can learn a kernel regression solution while in the rich regime. Our silent alignment phenomenon requires a separation of timescales between the evolution of the NTK's eigenfunctions and relative eigenvalues and a separate phase where the NTK grows only in scale. We demonstrate that, if these conditions are satisfied, than the final neural network function satisfies a representer theorem for the final NTK. We show analytically that these assumptions are realized in linear neural networks with small initialization trained on approximately whitened data and observe that the results hold for nonlinear networks and networks with multiple outputs. We demonstrate that silent alignment is highly sensitive to anisotropy in the input data.

Our results demonstrate that representation learning is not necessarily at odds with the learned neural network function being a kernel regression solution; i.e. a superposition of a kernel function on the training data. Further, by asking whether neural networks behave as kernel machine for some data-dependent kernel function, one can also shed light on their generalization properties (Bordelon et al., 2020; Canatar et al., 2021; Loureiro et al., 2021; Geiger et al., 2021) and transfer ability (see Appendix E).

REFERENCES

Kyle Aitken and Guy Gur-Ari. On the asymptotics of wide networks with polynomial activations. *ArXiv*, abs/2006.06687, 2020.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Anders Johan Andreassen and Ethan Dyer. Asymptotics of wide convolutional neural networks. *ArXiv*, abs/2008.08675, 2020.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 244–253. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/arora18a.html.

Sanjeev Arora, Simon Shaolei Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *NeurIPS*, 2019.

Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl8sJBYvH.

Aristide Baratin, Thomas George, César Laurent, R. Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *AISTATS*, 2021.

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.

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.

Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 11(1):1–13, 2020.

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.

Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine, 2020.

Simon Shaolei Du, Wei Hu, and J. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *NeurIPS*, 2018.

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.

Mario Geiger, Leonardo Petrini, and Matthieu Wyart. Landscape and training regimes in deep learning. *Physics Reports*, 924:1–18, 2021. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2021.04.001. URL https://www.sciencedirect.com/science/article/pii/S0370157321001290. Landscape and training regimes in deep learning.

B. Ghorbani, Song Mei, Theodor Misiakiewicz, and A. Montanari. Limitations of lazy training of two-layers neural networks. In *NeurIPS*, 2019.

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.

Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. 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. 4542–4551. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/huang20l.html.

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.

Arthur Jacot, François Ged, Franck Gabriel, Berfin Şimşek, and Clément Hongler. Deep linear networks dynamics: Low-rank biases induced by initialization scale and l2 regularization. *arXiv preprint arXiv:2106.15933*, 2021.

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.

Jaehoon Lee, Samuel S. Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *ArXiv*, abs/2007.15801, 2020.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks, 2021.

Bruno Loureiro, Cédric Gerbelot, Hugo Cui, Sebastian Goldt, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Capturing the learning curves of generic features maps for realistic data sets with a teacher-student model. *CoRR*, abs/2102.08127, 2021. URL https://arxiv.org/abs/2102.08127.

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 2021. doi: 10.1088/1742-5468/abf1f3. URL https://doi.org/10.1088/1742-5468/abf1f3.

Andrew M. Saxe, James L. Mcclelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*, 2014.

Haozhe Shan and Blake Bordelon. Rapid feature evolution accelerates learning in neural networks, 2021.

Dominik Stöger and Mahdi Soltanolkotabi. Small random initialization is akin to spectral learning: Optimization and generalization guarantees for overparameterized low-rank matrix reconstruction. *arXiv preprint arXiv:2106.15013*, 2021.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Jacob Abernethy and Shivani Agarwal (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 3635–3673. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

Greg Yang. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes, 2021.

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

Jacob A Zavatone-Veth, Abdulkadir Canatar, and Cengiz Pehlevan. Asymptotics of representation learning in finite bayesian neural networks. *arXiv preprint arXiv:2106.00651*, 2021.