Three Mechanisms of Feature Learning in an Analytically Solvable Model

Yizhou Xu YIZHOU.XU.CS@GMAIL.COM
Abdus Salam International Center for Theoretical Physics, Italy

Liu Ziyin ZIYINL@MIT.EDU
Massachusetts Institute of Technology, USA
NTT Research, USA

Abstract

We identify and exactly solve the learning dynamics of a one-hidden-layer linear model at any finite width whose limits exhibit both the kernel phase and the feature learning phase. Our solution identifies three novel prototype mechanisms of feature learning: (1) learning by alignment, (2) learning by disalignment, and (3) learning by rescaling. In sharp contrast, none of these mechanisms is present in the kernel regime of the model. We empirically demonstrate that these discoveries also appear in deep nonlinear networks in real tasks.

1. Introduction

It has been shown that for a neural network under certain types of scaling towards infinite width (or certain parameters), the learning dynamics can be precisely described by the neural tangent kernel (NTK) dynamics [22], or the "kernel regime". We say that a model is in the kernel regime if the NTK of the model remains unchanged throughout training, and the learning dynamics is linear in the model parameters. When the learning dynamics is not linear, we say that the model is in the feature learning regime. Since then, a lot of works have been devoted to the study of how the kernel evolves during training as it sheds light on nonlinear mechanisms of learning [5, 7, 8, 11, 16, 19, 20, 26, 30, 33]. However, despite the progress in understanding these infinite-width models, there have been limited results in understanding how feature learning actually happens and the relationship between these two limiting regimes. Arguably, the main theoretical gap is that until now, we do not know a single example of a finite-width exactly solvable model, whose NTK dynamics can be analytically and precisely described and exhibits both the NTK and feature learning regimes. In this paper, we analytically solve the evolution dynamics of a minimal finite-width model for arbitrary hyperparameters and initialization choices. The minimal model we address is a one-hidden-layer linear network. While the model is simple in nature, its loss landscape is nonconvex and its training involves strongly coupled dynamics, and the exact solution of its learning dynamics is previously unknown. Our results reveal three novel mechanisms of learning that are only existent in the feature learning phase of the network. Related literatures are reviewed in Appendix A. Proofs and additional theoretical results are in Appendix B. Additional experiments are in Appendix C.

2. An Exactly Solvable Model

Let us consider a two-layer linear network $f(\boldsymbol{x}) = \gamma \sum_{i=1}^{d} \sum_{j=1}^{d_0} u_i w_{ij} x_j$, where d_0 is the input size, d is the network width, $\boldsymbol{u}$ and $\mathbf{w}$ are the weight vector/matrix of the first and the second layer,

respectively, and γ is a normalization factor, which is essential for the mean-field scaling [8, 15, 35]. We consider that the network is trained on the MSE loss:

$$\tilde{L} = \mathbb{E}\left[(\gamma \sum_{i=1}^{d} \sum_{j=1}^{d_0} u_i w_{ij} \tilde{x}_j - y(\tilde{x}))^2 \right], \tag{1}$$

where we treated the target y as a function of $\tilde{x}$. Throughout this work, we write $\mathbb{E} := N^{-1} \sum_{\tilde{x}}$ to denote the averaging over the training set (although it does not have to be a finite sum). The training proceeds with the gradient flow algorithm. A little more general than the conventional study of NTK, we allow the two layers to have different learning rates, η_u and η_w:

$$\frac{du_i}{dt} = -\eta_u \frac{\partial \tilde{L}}{\partial u_i}, \quad \frac{dw_{ij}}{dt} = -\eta_w \frac{\partial \tilde{L}}{\partial w_{ij}}. \tag{2}$$

We restrict to when the data lies on a 1d manifold, and the following proposition shows that the learning dynamics under Eq.(1) is equivalent to that under a simplified loss.

Proposition 1 *Let $\tilde{x} = a\boldsymbol{n}$, where $a \in \mathbb{R}$ is a random variable and $\boldsymbol{n}$ is a fixed unit vector. Let $\boldsymbol{x} = \sqrt{\mathbb{E}[a^2]}\boldsymbol{n}$ and $y = \frac{\mathbb{E}[ay(\tilde{x})]}{\sqrt{\mathbb{E}[a^2]}}$. Then, the gradient flow of Eq.(1) equals the gradient flow of $L = \left[\gamma \sum_{i=1}^{d} \sum_{j=1}^{d_0} u_i w_{ij} x_j - y \right]^2$.*

Despite this being the simplest type of linear networks, its learning dynamics has not been analytically found in previous works. For this problem, only in two special settings, the dynamics of gradient descent (GD) have been solved. One is the standard kernel regime, where the dynamics are exactly linear [22]; the second case is when the two layers are initialized to be perfectly aligned, where u is a left eigenvector of w [10, 32]. The following theorem gives a precise characterization of the dynamics of u_i and w_i for arbitrary initialization and hyperparameter choices.

Theorem 2 *Let*

$$\begin{cases} p_i(t) := \frac{1}{2\rho}(\sqrt{\eta_u} \sum_{j=1}^{d_0} w_{ij}(t) x_j + \sqrt{\eta_w} \rho u_i(t)), \\ q_i(t) := \frac{1}{2\rho}(\sqrt{\eta_u} \sum_{j=1}^{d_0} w_{ij}(t) x_j - \sqrt{\eta_w} \rho u_i(t)), \end{cases} \tag{3}$$

where $\rho := \sqrt{\frac{1}{d_0} \sum_{i=0}^{d_0} x_i^2}$, and $P := \frac{1}{d} \sum_{j=1}^{d} p_j(0)^2$, $Q := \frac{1}{d} \sum_{j=1}^{d} q_j(0)^2$. If $P \neq 0$, then

$$\begin{cases} p_i(t) = p_i(0) \left[\frac{\alpha_+ + \xi(t)\alpha_-}{1 - \xi(t)} \right]^{1/2}, \\ q_i(t) = q_i(0) \left[\frac{\alpha_+ + \xi(t)\alpha_-}{1 - \xi(t)} \right]^{-1/2}, \end{cases} \tag{4}$$

where

$$\xi(t) := \frac{1 - \alpha_+}{1 + \alpha_-} \exp\left(-4t/t_c\right), \tag{5}$$

$$t_c := 1/\left(\sqrt{\eta_u \eta_w \gamma^2 \rho^2 y^2 + 4\rho^4 (\gamma^2 d)^2 PQ} \right), \tag{6}$$

$$\alpha_\pm := \frac{1}{2(\gamma^2 d)\rho^2 P} \left(\sqrt{\eta_u \eta_w} \gamma \rho y \pm t_c^{-1} \right). \tag{7}$$

Let us begin by analyzing each term and clarifying their meanings. In the theorem, we have transformed u_i and w_{ij} into an alternative basis p_i and q_i, and $\xi(t)$ is the only time-dependent term. Note that ξ decays exponentially towards zero at the time scale t_c. u_i and w_{ij} are obtained through

$$\begin{cases} u_i(t) = \frac{1}{\sqrt{\eta_w}}(p_i(t) - q_i(t)), \\ w_{ij}(t) = w_{ij}(0) + (p_i(t) - p_i(0) + q_i(t) - q_i(0))\,\frac{x_j}{\sqrt{\eta_u}\rho}. \end{cases}$$

The constants $\alpha_+\alpha_- = Q/P$ are two asymptotic scale factors. In the limit $t \to \infty$, we have that

$$p_i(\infty) = p_i(0)\sqrt{\alpha_+},\ q_i(\infty) = q_i(0)/\sqrt{\alpha_+}. \tag{8}$$

This directly gives us the mapping between the initialization to the converged solution. Unlike a strongly convex problem where the solution is independent of the initialization, we see that the converged solution for our model is strongly dependent on the initialization and on the choice of hyperparameters. Perhaps surprisingly, because $\alpha_\pm$ in (7) are functions of the learning rates, the converged solution (8) depends directly on the magnitudes of the learning rates. This directly tells us the implicit bias of gradient-descent training for this problem. Another special feature of the solution is that for any direction orthogonal to x, the model will remain unchanged during training. Let $m \perp x$, we have that $\sum_j w_{ij}(t)m_j = \sum_j w_{ij}(0)m_j$. Namely, the output of the model in the subspace where there is no data remains constant during training.

In the theorem, what is especially important is the characteristic time scale t_c, which is roughly the time it takes for learning to happen. Notably, the squared learning speed t_c^{-2} depends on two competing factors:

$$t_c^{-2} = \underbrace{\eta_u\eta_w\gamma^2\rho^2 y^2}_{\text{contribution from feature learning}} + \underbrace{4\rho^4(\gamma^2 d)^2 PQ}_{\text{contribution from kernel learning}}$$

The first factor depends on the input-output correlation and learning rate, which we will see is indicative of feature learning. The second term depends only on the input data and on the model initialization. We will see that when this term is dominant, the model is in the kernel regime. In fact, this result already invites a strong interpretation: the learning of the kernel regime is driven by the initialization and the input feature, whereas the learning in the feature learning regime is driven by the target mapping and large learning rates.

Using this theorem, one can compute the evolution of the NTK. Note that when different learning rates are used for different layers, the NTK needs to be defined slightly differently from the conventional definition. For the MSE loss, the NTK is the quantity K that enters the following dynamics: $\frac{df(x)}{dt} = 2K(x, x')(f(x') - y)$. This implies that for our problem,

$$K(x, x') = \gamma^2 x^T(\eta_w W^T W + \eta_u ||\boldsymbol{u}||^2 I)x', \tag{9}$$

according to Eq. (2) and Proposition 1. This definition agrees with the standard NTK if $\eta_w = \eta_u$.

3. Three Mechanisms of Feature Learning

We focus on a crucial effect predicted by this theorem, which differentiates it from previous results on similar problems. An important quantity our theory enables us to study is the evolution of the layer alignment $\zeta(t) := \boldsymbol{u}^T\boldsymbol{w}/||\boldsymbol{u}||||\boldsymbol{w}||$, which represents the cosine similarity between u and w.

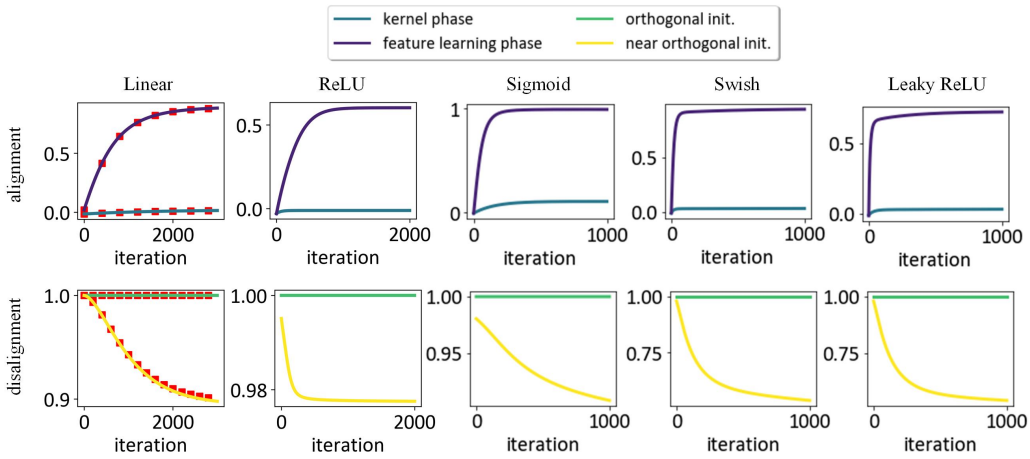


Figure 1: The evolution of ζ of two-layer networks with different settings. Specifically, we test linear, ReLU, sigmoid, swish, and leaky ReLU activations for both alignment (**upper**) and disalignment (**lower**) cases. For the linear network, we show the theoretical predictions obtained from (10) as lines and experimental results as points. The results for nonlinear networks are qualitatively similar.

Here, we set x to be 1d, because Theorem 2 suggests that the dynamics of GD training has only a rank-1 effect on the model. This quantity is especially interesting because it tells us how well-aligned the two layers are during training. Notably, this quantity vanishes as $d \to \infty$ if and only if the model is in the kernel regime, so it serves as a great metric for probing how feature learning happens.

Alignment and Disalignment. Let $x = 1$ and denoting $\alpha(t) = \frac{\alpha_+ + \xi\alpha_-}{1-\xi}$. By Theorem 2,

$$\zeta(t) = \frac{\alpha(t)P - Q/\alpha(t)}{\sqrt{(\alpha(t)P + Q/\alpha(t))^2 - (\frac{2}{d}\sum p_i q_i)^2}}, \tag{10}$$

where $4p_i q_i = u_i^2 - w_i^2 = const$ does not change during training. In general, the angle evolves by an $O(1)$ amount during training. In fact, the angle remains unchanged only in the orthogonal initialization case or in the kernel phase, where $\alpha(t) = 0/1$ throughout training (see Appendix C).

Let us first consider the kernel case. Here, the easiest way to see that ζ remains zero is to note that in the kernel regime, $\|u\|$ and $\|w\|$ are of order $\sqrt{d}$, whereas $u^T w$ is always of order $1/\gamma$. Therefore, $\zeta(t) = o(1)$ in the kernel phase (see Appendix B.3) and vanishes in the limit $d \to \infty$. Alternatively, one can see this from Theorem 2, which implies that in the kernel regime, $\alpha(t) = 1$ is a constant (see Appendix B.3), and in turn $\zeta(t) = 0$. Therefore, in the kernel regime, the two layers are essentially orthogonal to each other throughout training. This suggests one mechanism for the failure of the kernel learning phase. For a data point x, the hidden representation is wx, but predominant information in wx is ignored after the the layer u. This implies that the model will have a disproportionately larger norm than what is actually required to fit the data, which could in turn imply strong overfitting.

The second case is when the two layers are initialized in a parallel way. This setting is often called the "orthogonal initialization" [32]. In the orthogonal initialization, u is parallel to w, and so $p_i = Cq_i$ for a constant C. In this case, it is easy to very that $\zeta(t)^2 = 1$, meaning that u and w remain parallel or anti-parallel throughout training. Therefore, prior theory offers no clue regarding how ζ evolves in general.

Our solution implies a rather remarkable fact: ζ is always a monotonic function of t. To see this, its derivative is $\frac{d\zeta}{d\alpha} = \frac{(P+\frac{Q}{\alpha^2})(4PQ-(\frac{2}{d}\sum p_i q_i)^2)}{[(\alpha(t)P+Q/\alpha(t))^2-(\frac{2}{d}\sum p_i q_i)^2]^{3/2}}$. When u and w are parallel, this quantity is

zero, in agreement with our discussion about orthogonal initialization. When they are not parallel, we have that $4PQ - (\frac{2}{d}\sum p_i q_i)^2 > 0$ by the Cauchy inequality, and thus $d\zeta/d\alpha > 0$. Because $\alpha(t)$ monotonically evolves from 1 to $\alpha_+ > 0$, the evolution of ζ is also simple: $\zeta(t)$ monotonically increases if $\alpha_+ > 1$ or, equivalently, if

$$\frac{\sqrt{\eta_u \eta_w} y}{2\gamma d \rho P} + \sqrt{\left(\frac{\sqrt{\eta_u \eta_w} y}{2\gamma d \rho P}\right)^2 + \frac{Q}{P}} > 1 \tag{11}$$

and monotonically decreases if $\alpha_+ < 1$. ζ does not change if $\alpha_+ = 1$.

When does condition (11) hold? Let us focus on the case $y > 0$ because the theory is symmetric in the sign of y. The first observation is that it holds whenever $Q \geq P$, which is equivalent to $u^T(0)w(0) < 0$. Namely, if the model is making wrong predictions from the beginning, it will learn by aligning different layers. Moreover, this quantity also depends on the balance of the two learning rates. Notably, when the learning rates for the two different layers are the same, the change in ζ is independent of the learning rate. This implies that under the standard GD, the angle can be quite independent of the learning rates. However, the dependence on the learning rate becomes significant once we use different learning rates on the two layers. For example, when $\eta_u \gg \eta_w$ (or vice versa), this condition depends monotonically and (essentially) linearly on η_w, and making η_w close to η_u has the effect of making the two layers more aligned.

See Figure 1, where we show that the evolution of ζ between u and v of two-layer networks with $d = 10000$. It is trained on a regression task. Similar experimental results are observed for a classification task trained with the cross-entropy loss (Appendix C). We choose $\gamma = 1/\sqrt{d}$ for the kernel phase and $\gamma = 1/d$ for the feature learning phase. The initial weights are sampled from i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. Therefore, we have $P \approx Q$ and the initial $\zeta(0) \approx 0$. From (7) we have $\alpha_+ > 1$ if $y > 0$, so $\zeta(t)$ monotonically increases, but the increase is negligible in the kernel phase. Therefore, in this case, the model learns features by alignment. Meanwhile, the orthogonal initialization refers to the initialization scheme in [32], that is, ζ is initialized to be 1, so it remains 1 as predicted. In the near orthogonal case of Figure 1, we set $\zeta(0) \approx 1$ and the initial model output to be large. As predicted, $\zeta(t)$ monotonically decreases. In this case, the model learns by disalignment. From Figure 1, we also see that this phenomenon holds for all non-linear activation functions. The generalization of layer alignment and disalignment effects to higher dimension and deeper networks is given in Appendix C.

Learning by Rescaling. Learning can also happen by rescaling the output. The evolution of $||\boldsymbol{u}||$ and $||\boldsymbol{w}||$ are given by

$$||\boldsymbol{u}||^2 = d(\alpha P + \frac{Q}{\alpha}) - 2\sum_{i=1}^{d} p_i q_i, \; ||w||^2 = d(\alpha P + \frac{Q}{\alpha}) + 2\sum_{i=1}^{d} p_i q_i,$$

and, thus, $\frac{d||\boldsymbol{u}||^2}{d\alpha} = \frac{d||\boldsymbol{w}||^2}{d\alpha} = d(P - \frac{Q}{\alpha^2})$, which is positive when $\zeta > 0$, and negative when $\zeta < 0$. Thus, the rescaling coincides with the alignment, namely, $||\boldsymbol{u}||$ and $||\boldsymbol{w}||$ become larger when they are being aligned ($|\zeta|$ gets larger), and become smaller when they are being disaligned ($|\zeta|$ gets smaller). More explicitly, (1) $P > Q$ and $\alpha_+ > 1$, or $P < Q$ and $\alpha_+ < 1$, or $P = Q$: $||\boldsymbol{u}||$ and $||\boldsymbol{w}||$ monotonically increase. (2) $P > Q$ and $1 > \alpha_+ \geq \sqrt{Q/P}$, or $P < Q$ and $1 < \alpha_+ \leq \sqrt{Q/P}$: $||\boldsymbol{u}||$ and $||\boldsymbol{w}||$ and monotonically decrease. (3) $P > Q$ and $\alpha_+ < \sqrt{Q/P}$, or $P < Q$ and $\alpha_+ > \sqrt{Q/P}$: $||\boldsymbol{u}||$ and $||\boldsymbol{w}||$ first decrease, and then increase. (4) $\alpha_+ = 1$: everything keeps unchanged. Again, in the kernel phase, the scale change of the model vanishes. In the orthogonal initialization, however, this quantity changes by an $O(1)$ amount. Therefore, the orthogonal initialization essentially learns by rescaling the magnitude of the output.

References

[1] Anders Andreassen and Ethan Dyer. Asymptotics of wide convolutional neural networks. *arXiv preprint arXiv:2008.08675*, 2020.

[2] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International conference on machine learning*, pages 244–253. PMLR, 2018.

[3] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

[5] Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. *arXiv preprint arXiv:2111.00034*, 2021.

[6] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[7] Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2021.

[8] Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *Advances in Neural Information Processing Systems*, 35:32240–32256, 2022.

[9] Blake Bordelon and Cengiz Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Lukas Braun, Clémentine Dominé, James Fitzgerald, and Andrew Saxe. Exact learning dynamics of deep linear networks with prior knowledge. *Advances in Neural Information Processing Systems*, 35:6615–6629, 2022.

[11] Shuxiao Chen, Hangfeng He, and Weijie Su. Label-aware neural tangent kernel: Toward better generalization and local elasticity. *Advances in Neural Information Processing Systems*, 33: 15847–15858, 2020.

[12] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

[13] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[14] Kenji Fukumizu. Effect of batch learning in multilayer neural networks. *Gen*, 1(04):1E–03, 1998.

[15] 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, 2020.

[16] Mario Geiger, Leonardo Petrini, and Matthieu Wyart. Landscape and training regimes in deep learning. *Physics Reports*, 924:1–18, 2021.

[17] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, 31, 2018.

[18] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *arXiv preprint arXiv:1909.05989*, 2019.

[19] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *International conference on machine learning*, pages 4542–4551. PMLR, 2020.

[20] Wei Huang, Weitao Du, and Richard Yi Da Xu. On the neural tangent kernel of deep networks with orthogonal initialization. *arXiv preprint arXiv:2004.05867*, 2020.

[21] Dongsung Huh. Curvature-corrected learning dynamics in deep neural networks. In *International Conference on Machine Learning*, pages 4552–4560. PMLR, 2020.

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[23] Andrew K Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv preprint arXiv:1809.10374*, 2018.

[24] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019.

[25] Yuanzhi Li, Tengyu Ma, and Hongyang R Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In *Conference on learning theory*, pages 2613–2682. PMLR, 2020.

[26] Chaoyue Liu, Libin Zhu, and Misha Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33:15954–15964, 2020.

[27] 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.

[28] Franco Pellegrini and Giulio Biroli. An analytic theory of shallow networks dynamics for hinge loss classification. *Advances in Neural Information Processing Systems*, 33:5356–5367, 2020.

[29] Huy Tuan Pham and Phan-Minh Nguyen. Limiting fluctuation and trajectorial stability of multilayer neural networks with mean field training. *Advances in Neural Information Processing Systems*, 34:4843–4855, 2021.

[30] Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Feature learning in neural networks and kernel machines that recursively learn features. *arXiv preprint arXiv:2212.13881*, 2022.

[31] Daniel A Roberts, Sho Yaida, and Boris Hanin. *The principles of deep learning theory*, volume 46. Cambridge University Press Cambridge, MA, USA, 2022.

[32] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[33] James B Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. *arXiv preprint arXiv:2303.15438*, 2023.

[34] Salma Tarmoun, Guilherme Franca, Benjamin D Haeffele, and Rene Vidal. Understanding the dynamics of gradient flow in overparameterized linear models. In *International Conference on Machine Learning*, pages 10153–10161. PMLR, 2021.

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

[36] Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

[37] Liu Ziyin, Mingze Wang, and Lei Wu. The implicit bias of gradient noise: A symmetry perspective. *arXiv preprint arXiv:2402.07193*, 2024.

Appendix A. Related Work

Kernel and feature learning. Due to the importance of the NTK in understanding the training and generalization of neural networks, a lot of works are devoted to understanding its structure and when and how it changes as training proceeds. Under the NTK scaling, it is shown that NTK remains unchanged in the infinite-width limit [4, 22, 24], where the network is asymptotically equivalent to the kernel regression using NTK. Higher order feature learning corrections of the NTK have also been studied [1, 13, 18, 31]. An important alternative to the NTK parameterization is the mean-field (or μP) parameterization where features evolve at infinite width [8, 27, 35]. Within this literature, the works closest to ours are those computing finite width corrections [9, 28, 29]. However, these results are perturbative in nature and applicable when the width is large. Our study has the same goal of understanding the learning dynamics but with a different approach. We solve an analytically solvable model that admits analysis both when the model size is finite and infinite.

Deep linear networks. Our work is also related to the analysis of deep linear networks, which has provided significant insights into the loss landscape [6, 14], optimization [10, 21, 32, 34], generalization [17, 23] and learning dynamics[2, 3, 37] of neural networks. Closely related to ours are Refs. [5, 10, 32], which solve the learning dynamics of linear models under special initializations. Our main advancement in this respect is to exactly solve the learning dynamics from arbitrary initializations, which we then utilize to analyze the effect of the initialization scale for learning.

Appendix B. Theoretical Concerns

B.1. Proof of Proposition 1

Proof To make the analysis more concrete, we consider the standard loss function $\tilde{L}(u,w) = \frac{1}{N}\sum_{k=1}^{N}(\gamma\sum_{i=1}^{d}\sum_{j=1}^{d_0}u_i w_{ij}\tilde{x}_{jk} - \tilde{y}_k)^2$, where N is the size of the training set.

Data points lie in a 1d-subspace, meaning that $\tilde{x}_{jk} = a_k n_j$ for a constant unit vector $\boldsymbol{n}$. Because of the 1d nature of the data, the training dynamics on this loss function is completely identical to training on the following loss $L(u,w) = (\gamma\sum_{i=1}^{d}\sum_{j=1}^{d_0}u_i w_{ij}x_j - y)$, where $x_j = \sqrt{\sum_{k=1}^{N}a_k^2}n_j$ and $y = \frac{\sum_{k=1}^{N}a_k\tilde{y}_k}{\sqrt{\sum_{k=1}^{N}a_k^2}}$. This is because

$$\begin{aligned}\tilde{L}(u,w) &= (\sum_{k=1}^{N}a_k^2)(\gamma\sum_{i=1}^{d}\sum_{j=1}^{d_0}u_i w_{ij}n_j)^2 - 2(\sum_{k=1}^{N}a_k\tilde{y}_k)(\gamma\sum_{i=1}^{d}\sum_{j=1}^{d_0}u_i w_{ij}n_j) + \sum_{k=1}^{N}y_k^2 \\ &= L(u,w) + \sum_{k=1}^{N}y_k^2 - y^2.\end{aligned} \tag{12}$$

Therefore, without loss of generality, the training on the standard loss $\tilde{L}$ is identical to the training on L because the difference is only by a constant that does not affect gradient descent training. This setting is thus equivalent to the case when the dataset contains only a single data point $(\boldsymbol{x}, y)$.[1] As is clearly shown from this example, using the notation in terms of x and y is much simpler to understand than using $\tilde{x}_{jk}$ and y_k. We believe that this notation is necessary and greatly facilitates the later discussions once the readers accept it.

1. Essentially, this is because we only need two points to specify a line. Also, it is trivial to extend to the case when y is a vector that spans only a one-dimensional subspace.

Finally, all these notations can also be written in terms of $\mathbb{E}_x := \frac{1}{N}\sum_{k=1}^{N}$, which is the notation we chose for introducing the lemma.

■

B.2. Proof of Theorem 2

Proof By the definition of the gradient flow algorithm,

$$\begin{aligned}\frac{du_i}{dt} &= -\eta_u \frac{\partial L}{\partial u_i} = -2\eta_u \gamma \sum_{j=1}^{d_0} w_{ij}x_j \left(\gamma \sum_{i=1}^{d}\sum_{j=1}^{d_0} u_i w_{ij} x_j - y\right), \\ \frac{dw_{ij}}{dt} &= -\eta_w \frac{\partial L}{\partial w_{ij}} = -2\eta_w \gamma u_i x_j \left(\gamma \sum_{i=1}^{d}\sum_{j=1}^{d_0} u_i w_{ij} x_j - y\right),\end{aligned} \tag{13}$$

which implies the following two conservation laws:

$$\frac{d}{dt}(\eta_u \sum_{j=1}^{d_0} w_{ij}^2 - \eta_w u_i^2) = 0, \tag{14}$$

$$\frac{d}{dt}\left(\frac{w_{ij}}{x_j} - \frac{w_{ij'}}{x_{j'}}\right) = \frac{1}{x_j}\frac{dw_{ij}}{dt} - \frac{1}{x_{j'}}\frac{dw_{ij'}}{dt} = 0. \tag{15}$$

From Eq. (13), we can denote $\frac{dw_{ij}}{dt} = u_i x_j A$, which implies

$$\begin{aligned}&\frac{d}{dt}\left(\sum_{j=1}^{d_0} w_{ij}^2 - \frac{1}{\sum_{j=1}^{d_0} x_j^2}(\sum_{j=1}^{N} w_{ij}x_j)^2\right) \\ &= 2\sum_{j=1}^{d_0} u_i w_{ij} x_j A - 2\frac{\sum_{j=1}^{d_0} u_i w_{ij} x_j}{\sum_{j=1}^{d_0} x_j^2}\sum_{j=1}^{d_0} x_j^2 A = 0.\end{aligned} \tag{16}$$

Now we denote $p_i(t) := \frac{1}{2\rho}(\sqrt{\eta_u}\sum_{j=1}^{d_0} w_{ij}(t)x_j + \sqrt{\eta_w}\rho u_i(t))$ and $q_i(t) \equiv:= \frac{1}{2\rho}(\sqrt{\eta_u}\sum_{j=1}^{d_0} w_{ij}(t)x_j - \sqrt{\eta_w}\rho u_i(t))$, and thus

$$p_i(t)q_i(t) = \frac{1}{4\rho^2}\left(\eta_u(\sum_{j=1}^{N} w_{ij}(t)x_j)^2 - \eta_w \rho^2 u_i(t)^2\right). \tag{17}$$

Take derivatives on both sides and use (14) and (16). Then we have

$$\begin{aligned}\frac{d}{dt}\left(p_i(t)q_i(t)\right) &= \frac{1}{4}\frac{d}{dt}\left(\frac{\eta_u}{\sum_{j=1}^{d_0} x_j^2}(\sum_{j=1}^{N} w_{ij}x_j)^2 - \eta_w u_i^2\right) \\ &= \frac{1}{4}\frac{d}{dt}\left(\eta_u \sum_{j=1}^{d_0} w_{ij}^2 - \eta_w u_i^2\right) = 0\end{aligned}. \tag{18}$$

Further, substituting (13) into the definition of p_i and q_i, we have

$$\frac{dp_i}{dt} = -2\gamma\sqrt{\eta_u\eta_w}p_i\rho\left(\sum_{j=1}^{d}(p_j^2 - q_j^2)\frac{\gamma\rho}{\sqrt{\eta_u\eta_w}} - y\right). \tag{19}$$

(19) implies $\frac{1}{p_i}\frac{dp_i}{dt} = \frac{1}{p_{i'}}\frac{dp_{i'}}{dt}$, further leading to another conservation law

$$\frac{d}{dt}\frac{p_i(t)}{p_{i'}(t)} = 0. \tag{20}$$

for all $i, i' = 1, 2, \cdots, d$. Then according to (18) and (20), we have $p_{i'}(t) = p_{i'}(0)\frac{p_i(t)}{p_i(0)}$ and $q_{i'}(t) = q_{i'}(0)\frac{p_{i'}(0)}{p_{i'}(t)}$. Substituting them into (19), and we obtain a differential equation with only one variable p_i

$$\frac{dp_i}{dt} = -2p_i\left(\frac{(\gamma^2 d)\rho^2 P}{p_i(0)^2}p_i^2 - \frac{(\gamma^2 d)\rho^2 Q p_i(0)^2}{p_i^2} - \gamma\rho y\sqrt{\eta_u\eta_w}\right), \tag{21}$$

where

$$P = \frac{1}{d}\sum_{i=1}^{d}p_i(0)^2,\ Q = \frac{1}{d}\sum_{i=1}^{d}q_i(0)^2. \tag{22}$$

This differential equation is analytically solvable by integration

$$t = -\int_{p_i(0)^2}^{p_i^2}\frac{d\zeta}{4\left(\frac{(\gamma^2 d)\rho^2 P}{p_i(0)^2}\zeta^2 - \gamma xy\sqrt{\eta_u\eta_w}\zeta - (\gamma^2 d)\rho^2 Q p_i(0)^2\right)} \tag{23}$$

Because the denominator as a quadratic polynomial has two different roots $\alpha_\pm$, the result of the integration is

$$t = -\frac{t_c}{4}\log\frac{p_i(t)^2/p_i(0)^2 - \alpha_+}{p_i(t)^2/p_i(0)^2 - \alpha_-} + const, \tag{24}$$

leading to

$$\frac{p_i(t)^2/p_i(0)^2 - \alpha_+}{p_i(t)^2/p_i(0)^2 - \alpha_-} = \frac{1-\alpha_+}{1-\alpha_-}\exp\left(-4t/t_c\right), \tag{25}$$

which gives (4). ■

Proposition 3 *Under the condition in Theorem 2, if $P = 0$ and $Q \neq 0$, the result becomes*

$$p_i(t) = 0 \tag{26}$$

$$q_i(t) = q_i(0)\sqrt{\frac{\alpha'\xi'(t)}{1-\xi'(t)}} \tag{27}$$

where

$$\xi'(t) := \frac{1}{1+\alpha'}\exp\left(-4\sqrt{\eta_u\eta_w}\gamma\rho y t\right), \tag{28}$$

and

$$\alpha' := \frac{\sqrt{\eta_u \eta_w}\gamma\rho y}{(\gamma^2 d)\rho^2 Q}. \tag{29}$$

Specially, if $P = Q = 0$, we have $p_i(t) = q_(t) = 0$, so the gradient flow will be stuck at the trivial saddle point.

The proof is similar to the proof of Theorem 2, because we can similarly obtain

$$\frac{dq_i}{dt} = -2q_i \left(\frac{(\gamma^2 d)\rho^2 Q}{q_i(0)^2} q_i^2 + \gamma\rho y \sqrt{\eta_u \eta_w} \right). \tag{30}$$

Its solution gives Proposition 3.

We note that the behavior of the solution is quite different from $P \neq 0$: when $y \leq 0$, we can obtain a solution with zero loss in the end, but when $y > 0$, the gradient flow will converge to the trivial saddle point $p_i = q_i = 0$.

B.3. Phase Diagrams

Our theory can be applied to study the learning of different scaling limits, where we scale the hyperparameters with a scaling parameter κ towards infinity. Here, κ is an abstract quantity that increases linearly, and all the hyperparameters including the width are a power-law function of κ. Conventionally, the choice of κ is the model width; however, this excludes the discussion of the lazy training regime in the theory, where the model width is kept fixed and the scaling parameter is the model output scale γ.

We first establish the necessary and sufficient condition for learning to happen: the learning time t_c needs to be of order $\Theta(1)$. When it diverges, learning is stationary and frozen at initialization. When it vanishes to zero, the discrete-time SGD algorithm will be unstable, a point that is first pointed out by [35]. Therefore, we first study the condition for t_c to be of order 1, which is equivalent to the condition that (assuming x, y are order 1)

$$\eta_u \eta_w \gamma^2 + (\gamma^2 d)^2 PQ = \Theta(1). \tag{31}$$

For Gaussian initialization $u_{i0} \sim \mathcal{N}(0, \sigma_u^2)$ and $w_{i0} \sim \mathcal{N}(0, \sigma_w^2)$, P and Q are random variables with expectation $(\eta_w \sigma_u^2 + \eta_u \sigma_w^2)/4$ and variance $(\eta_w \sigma_u^2 + \eta_u \sigma_w^2)^2/8d$. Generally, all hyperparameters are powers of κ: $d \propto \kappa^{c_d}$, $\gamma \propto \kappa^{c_\gamma}$, $\sigma_w^2 \propto \kappa^{c_w}$, $\sigma_u^2 \propto \kappa^{c_u}$, $\eta_w \propto \kappa^{c_{\eta_w}}$ and $\eta_u \propto \kappa^{c_{\eta_u}}$. For simplicity, we set the input dimension d_0 to be a constant.

Equation (31) implies

$$\max\left\{2c_\gamma + c_{\eta_u} + c_{\eta_w},\ 2c_\gamma + c_d + \max\{c_{\eta_w} + c_u, c_{\eta_u} + c_w\}\right\} = 0. \tag{32}$$

Whatever choice of the exponents that solves the above equation is a valid learning limit for a neural network. The phase of the network depends on the relative order of the above two terms.

Definition 4 *A model is in the kernel phase if (1) Eq. (9) is independent of t as $\kappa \to \infty$ (2) $NTK = \Theta(1)$.*

When $t_c = \Theta(1)$, a model is said to be in the feature learning phase if it is not in the kernel phase.

Theorem 5 *When Eq. (32) holds, a model is in the kernel phase if and only if* $\lim_{\kappa\to\infty} P/Q = 1$, *a.s.. and*

$$c_d + \max\{c_{\eta_w} + c_u, c_{\eta_u} + c_w\} > c_{\eta_u} + c_{\eta_w}. \tag{33}$$

Proof Using $P/Q = 1$ and (33), we have

$$\lim_{\kappa\to\infty} \alpha_+ = \lim_{\kappa\to\infty} \frac{2\rho^2(\gamma^2 d)\sqrt{PQ}}{2\rho^2(\gamma^2 d)P} = 1.\ a.s. \tag{34}$$

By definition, $\xi(t)$ is a monotonic function. As $\frac{\alpha_+ - \xi\alpha_-}{1-\xi} = \frac{\alpha_+ - \alpha_-}{1-\xi} + \alpha_-$ is monotonous to ξ, it evolves from 1 to α_+ monotonously. Then according to Equation (9), $\lim_{\kappa\to\infty} K(x, x')(t) = \lim_{\kappa\to\infty} K(x, x')(0)$ if and only if $\lim_{\kappa\to\infty} \alpha_+ = 1$.

From Equation (32), Equation (33) also implies

$$2c_\gamma + c_d + \max\{c_{\eta_w} + c_u, c_{\eta_u} + c_w\} = 0. \tag{35}$$

Therefore, we can see that the NTK remains $\Theta(1)$ because

$$\gamma^2 d\alpha_+ P = \Theta(\kappa^{2c_\gamma + c_d + \max\{c_{\eta_w} + c_u, c_{\eta_u} + c_w\}}) = \Theta(1). \tag{36}$$

The proof is complete. ■

The necessary condition $P/Q = 1$ for the model being in the kernel phase is interesting and highlights the important role of initialization in deep learning. There are three common cases when this holds:
1. $d \to \infty$ and u_0 and w_0 are independent (standard NTK);
2. d is finite and the initial model output is zero: $\sum_{i=1}^{d}\sum_{j=1}^{d_0} u_i w_{ij} x_j = 0$ (lazy training)
3. d is finite, $\kappa \to \infty$ and $c_u + c_{\eta_w} \neq c_w + c_{\eta_u}$;

The first case is the standard way of initialization, from which one can derive the classic analysis of the kernel phase by invoking the law of large numbers. The second case is the assumption used in the lazy training regime [12]. ([12] assumes $c_\gamma = 1$, $c_{\eta_u} = c_{\eta_w} = -2$ and $c_u = c_w = 0$, satisfying the conditions of the exponents (32) and (33).) This case, however, relies on a special initialization, and thus our results better illustrate the occurrence of the kernel phase for advanced initialization methods where different weights can be correlated. The third case happens when the learning rate and the initialization are not balanced. This suggests that to achieve feature learning, one should make sure that the learning rate and the initialization are well balanced: $c_u = c_w$.

In conclusion, the overall phase is (1) **kernel phase**, if the first term in (32) is strictly smaller than the second term: $0 = c_\gamma + c_d + \max\{c_{\eta_w} + c_u, c_{\eta_u} + c_w\} > c_{\eta_u} + c_{\eta_w}$ and $\lim_{\kappa\to\infty} P/Q = 1$, (2) **feature learning phase** if otherwise. A key difference between these two phases is whether the evolution of the NTK is $O(1)$, or equivalently whether the model learns features.

An intriguing fact is that layers tend to align in the feature learning phase. Suppose $P \approx Q$ and $\sum_{i=1}^{d} p_i q_i \approx 0$, which holds for Gaussian initialization and d sufficiently large, (10) leads to $\zeta(t) \approx \frac{\alpha(t)^2 - 1}{\alpha(t)^2 + 1}$, which monotonously changes from 0 to $\frac{\alpha_+^2 - 1}{\alpha_+^2 + 1}$. In the feature learning phase, two terms in (6) are of the same order, so we can suppose $\sqrt{\eta_u \eta_w}\gamma\rho y \geq 2K\rho^2\gamma^2 dP$ without loss of generality, where K is a certain constant. This further leads to a non-zero upper bound of the alignment $\zeta(\infty) \geq \frac{(K+1)^2 - 1}{(K+1)^2 + 1}$, in clear contrast to the kernel phase, where the alignment remains asymptotically zero for Gaussian initialization.

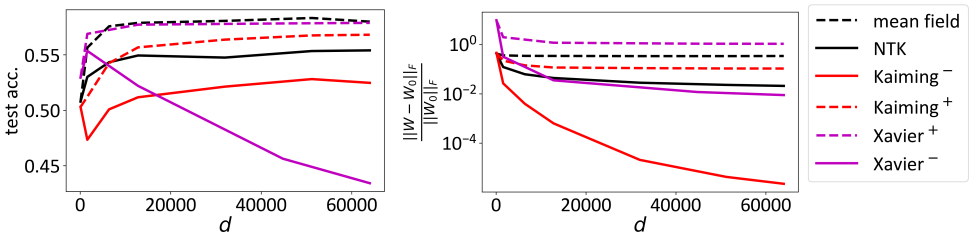


Figure 2: A two-layer fully connected ReLU net with d neurons trained on the CIFAR-10 dataset for 10000 epochs with batch size 128. The kernel phase is shown in solid lines and the feature learning phase is shown in dashed lines. As the theory predicts, both types of initialization can be turned into either the feature learning or the kernel phase by choosing different combinations of γ and η. **Left**: the best test accuracy during training. **Right**: relative distance from the initialization.

Table 1: Phases of learning in different scaling limits. For brevity, the learning rates of the two layers are set to be equal. The first block shows that the models can be frozen or unstable if we do not scale η accordingly. The second block shows that one can always choose η such that the model training is stable and does not freeze. The third and fourth blocks show that one can always choose a pair of η and γ such that the model is either in the feature learning phase or the kernel phase.

scaling	NTK	Mean Field [27]	Xavier init.	Kaiming init.	lazy [12]
c_d	1	1	1	1	0
c_γ	-1/2	-1	0	0	1
c_u	0	0	-1	0	0
c_w	0	0	-1	-1	0
c_η	0	0	0	0	0
phase	kernel	frozen	learning	unstable	unstable
c_η^*	0	1	0	-1	-2
phase	kernel	learning	learning	kernel	kernel
c_η^+	1	1	0	1	0
c_γ^+	-1	-1	0	-1	0
phase	learning	learning	learning	learning	learning
c_η^-	0	0	-2	-1	-2
c_γ^-	-1/2	-1/2	1	0	1
phase	kernel	kernel	kernel	kernel	kernel

B.3.1. PHASES DIAGRAM OF INFINITE-WIDTH MODELS

Now, let us focus on the case when $\kappa = d \to \infty$ (thus $c_d = 1$), corresponding to the infinite width limit considered in the NTK and feature learning literature [22, 25, 35]. In this limit, $\lim_{\kappa\to\infty} P/Q = 1$ naturally holds by law of large numbers. Therefore, a sufficient and necessary condition for the kernel phase is $0 = c_\gamma + c_d + \max\{c_{\eta_w} + c_u, c_{\eta_u} + c_w\} > c_{\eta_u} + c_{\eta_w}$.

The following corollaries are direct consequences of Eq. (32).

Corollary 6 *For any c_γ, c_u and c_w, choosing $c_{\eta_u} = c_{\eta_w} = \min\{-c_\gamma, -2c_\gamma - c_d - \max\{c_u, c_w\}\}$ ensures that the model is stable.*

Corollary 7 *For any c_u and c_w, choosing $c_{\eta_u} = c_{\eta_w} = c_\eta$ and $c_\gamma = -c_\eta$ with $c_\eta \geq c_d + \max\{c_u, c_w\}$ leads to a feature learning phase.*

Corollary 8 *For any c_u and c_w, choosing $c_\gamma = -\frac{1}{2}(c_d + \max\{c_u, c_w\} + c_\eta)$ and $c_\eta < c_d + \max\{c_u, c_w\}$ leads to a kernel phase.*

They imply two important messages: for every initialization scheme, (1) one can choose an optimal learning rate such that the learning is stable; (2) one can choose an optimal pair of learning rate and output scale γ such that the model is in the feature learning phase. Point (1) agrees with the analysis in [35], whereas point (2) is a new insight we offer. See Table 1 for the classification of different common scalings. We choose scalings according to Corollary 7 and 8, to turn each model into the feature learning or the kernel phase.

See Figure 2. We implement a two-layer FCN on the CIFAR-10 dataset with ReLU activation. We run experiments with the scalings of the standard NTK, standard mean-field, Kaiming model, and Xavier model. c_γ and c_η are chosen according to Table 1. Here, we use the superscript $+$ to denote the type of scaling that leads to a feature learning phase, and $-$ denotes the kernel phase. For the Kaiming and Xavier model, we choose both $c_\eta^\pm$ and $c_\gamma^\pm$, and refer them as Kaiming$^\pm$ and Xavier$^\pm$, respectively. The left figure shows that turning the Kaiming model into the feature learning phase improves the test accuracy by approximately 5%, similar to the gap between the standard NTK model and the mean-field model. Meanwhile, turning the Xavier model into the kernel phase decreases the test accuracy by approximately 10%. This is because the fixed kernel restricts the generalization ability in the kernel phase, and the difference between these models in the kernel phase might be attributed to their different kernels. Thus, in agreement with the theory, choosing different combinations of the output scale γ and η can turn any initialization into the feature learning phase. This insight could be very useful in practice, as the Kaiming init. is predominantly used in deep learning practice and is often observed to have better performance at common widths of the network. Our result thus suggests that it is possible to keep its advantage even if we scale up the network. Further, our results also imply that any valid learning regimes transfer well when the model gets larger, while in the feature learning phase, a larger model generally leads to better performance. This is consistent with earlier work [36].

B.3.2. PHASE DIAGRAM FOR INITIALIZATIONS

Now, we study the case when d is kept fixed, while other variables scale with $\kappa \to \infty$. In this case, the phase diagram is also given by Theorem 5. See Figure 3 for an experiment. We set $c_u = \max\{0, c_w\}$ and $c_\gamma = \min\{-c_w/2, 0\}$. This choice ensures that the initial model output is $O(1)$. In this case, (32) is satisfied. By Theorem 5, the network is in the kernel phase if and only if $c_w > 0$. One important example for this section is the lazy training regime, where $c_u = c_w = c_d = 0$, and we can choose $c_\gamma = 1$ and $c_\eta = -2$ according to Corollary 8, leading to a kernel phase in finite width.

Another example is to consider large initialization, i.e., $c_u = c_w = c > 0$. In this case, we can choose $c_\eta = c$ and $c_\gamma = -c$ according to Corollary 7, leading to a feature learning phase. Actually, this choice of the normalization factor γ cancels out the scaling of the initialization. On the other hand, if we choose $\gamma = 1$ as commonly done, we have to choose $c_\eta = -c$ according to Corollary 8, leading to a kernel space. This might be another possible explanation that larger initialization often leads to worse performance empirically. Like before, we implement a two-layer fully connected ReLU network on the CIFAR-10 dataset with $d = 2000$. We choose $\kappa = 10$ for illustration purposes. A clear distinction is observed between the feature learning phase and the kernel phase. (1) In

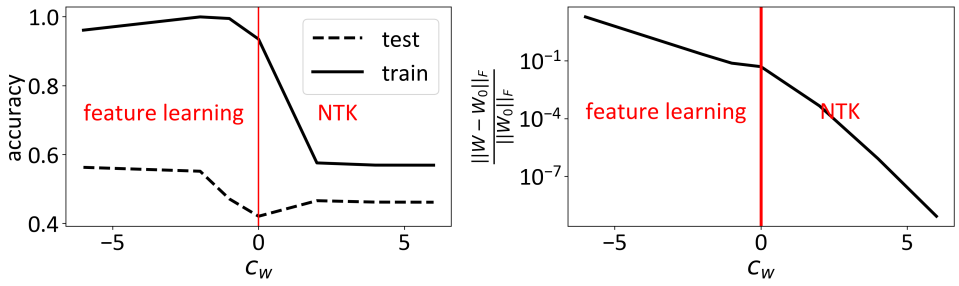
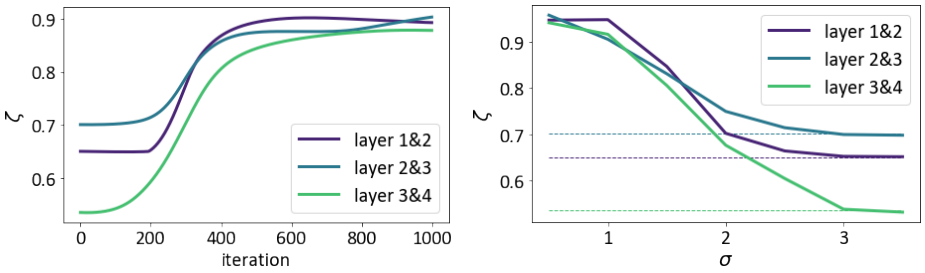


Figure 3: A two-layer FCN with different initialization scales trained on the CIFAR-10 dataset. We see that finite-width models can also exhibit qualitative differences between the feature learning and the kernel phases when other hyperparameters are scaled toward infinity. Notably, this scaling is different from the lazy training scaling, implying that there are numerous (actually infinitely many) ways for the model to enter the kernel phase, even at a finite width.



(*a*) alignment in a 4-layer FCN (*b*) alignment vs. initialization scale

Figure 4: The alignment angle ζ between different layers of a four-layer FCN with ReLU activation trained on MNIST. (b) shows the final alignment for different initialization scale σ, while (a) shows training curves corresponding to $\sigma = 1$. The dashed lines in (b) show the initial alignment.

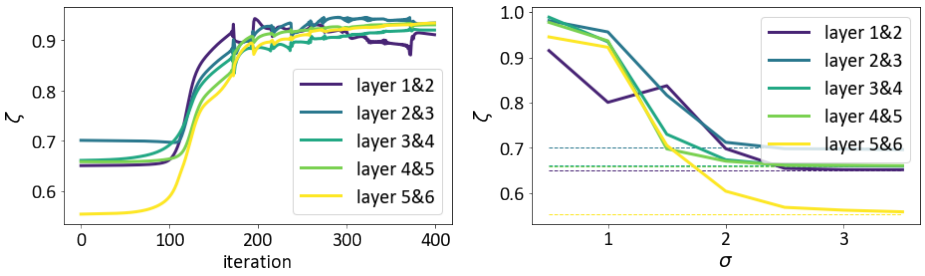
Figure 3(a), the training accuracy can reach 1.0 in the feature learning phase but not the kernel phase, because the NTK in the kernel phase is fixed, and thus the best training accuracy is limited by the fixed kernel. (2) As discussed in the previous section, the test accuracy In Figure 3(a) is about 5% higher in the feature learning phase due to its trainable kernel. (3) In Figure 3(b), the weight matrices evolve significantly in the feature learning phase but not the kernel phase.

Appendix C. Additional Experimental Concerns

C.1. Additional Experiments in Section 3

In Figure 1, we choose $x = 1, y = 2$, and for others we randomly sample 100 points from $\mathcal{N}(0, 1)$ as data points x, and set $y = 2x + \mathcal{N}(0, 1)/10$ as the target. The learning rates are chosen such that the model converges well within given iterations. For the orthogonal initialization, we initialize the model as $u \sim \mathcal{N}(0, 10I_d)$ and $w \sim u + \mathcal{N}(0, 0.1I_d)$.

To generalize layer alignment and disalignment effects to higher dimension and deeper networks, we define $\zeta := \frac{||UW||}{||U||||W||}$, where U, W are the weight matrices of two consecutive layers and the L2 norm for matrices is used. See Figure 4 for a four-layer fully connected network (FCN) with ReLU



(*a*) alignment in a 6-layer FCN (*b*) alignment vs. initialization scale

Figure 5: The alignment angle ζ between different layers of a six-layer FCN trained on MNIST, with the same settings as Figure 4.

activation and different initialization scales trained on MNIST datasets. Figure 4 (a) shows that the alignment between consecutive layers increases during training, and Figure 4 (b) demonstrates that layers stop being aligned for large initialization. These results are consistent with simpler settings, verifying that our analysis applies to deeper networks with non-linear activation.

In Figure 4, to avoid the implicit bias of SGD to make layers aligned [37], we consider full-batch GD with batch size 2000 and constant learning rate. The learning rates are chosen separately for each model such that the model converges well in 1000 iterations, with training accuracy above 95%. All models use the standard Kaiming initialization, but we scale each layer by σ. The results in Figure 4 also extend to deeper networks, although the training dynamics of deeper FCNs are less stable, as shown in Figure 5.

Moreover, we observe qualitatively the same phenomenon for all kinds of activation functions in the classification task in Figure 6, where the task is to classify training samples from $\mathcal{N}(0,1)$ and $\mathcal{N}(4,1)$. Initialization is the same as in Figure 1, but the binary cross-entropy loss is used. From Figure 6 we can also see that layers tend to align in the feature learning regime when they are initialized to be disaligned, and vice versa. Note that because of the binary cross-entropy loss, ζ keeps decreasing even after the loss converges. Further, because of the binary cross-entropy loss, ζ deviates from one for non-linear activation functions other than ReLU.

In Figure 7, we train a Resnet18 network on the CIFAR-10 dataset with hyperparameters borrowed from https://github.com/kuangliu/pytorch-cifar. The only difference is that we scale each layer by σ and record the test accuracy together with the sum of the norm of all layers. Figure 7 shows that a larger initialization leads to worse performance. Note that this example can only be explained through the disalignment effect because (1) the model achieves 100% train accuracy in all settings, yet (2) a larger initialization leads to a larger norm at the end of the training, which also correlates with worse performance. Another piece of evidence is the commonly observed underperformance of kernel models. In the kernel phase, the model norm diverges and the model alignment is always zero, which could be a hint of strong overfitting. Therefore, our theory suggests that it would be a great idea for future works to develop algorithms that maximize layer alignment while minimizing the change in the output scale.

C.2. Experiments in Section B.3.1

In Section B.3.1, we utilize a two-layer FCN with the ReLU activation and d hidden units. The input is vectorized and normalized, so the input dimension is $d_0 = 3072$. The cross-entropy loss and the

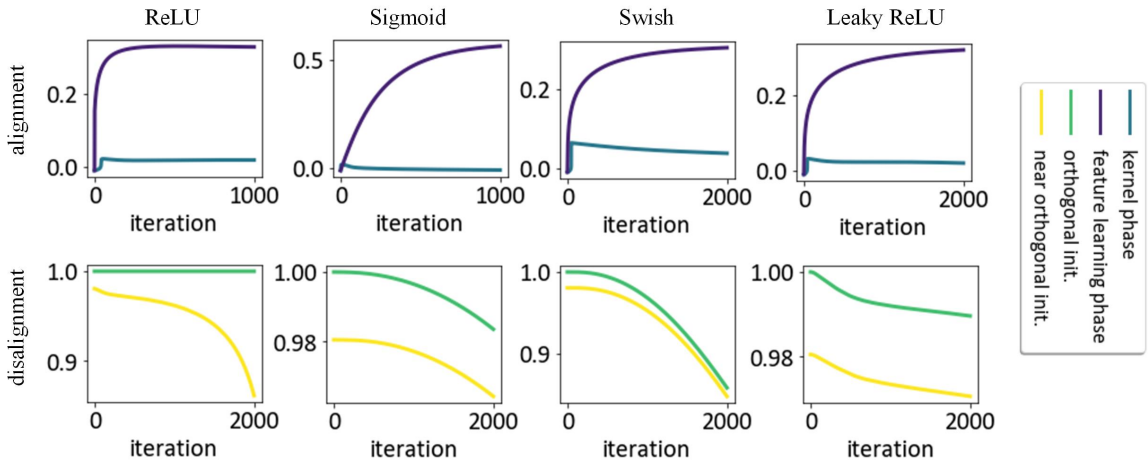


Figure 6: The evolution of the alignment angle ζ between u and v across two-layer ReLU, sigmoid, swish, and leaky ReLU networks with $d = 10000$. The task is to classify two Gaussian distributions.

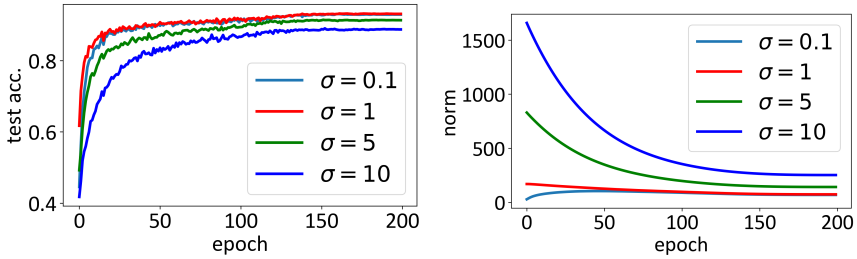


Figure 7: The initialization scale σ correlates negatively with the performance of Resnet-18 on the CIFAR-10 dataset. **Left**: test accuracy. Here, σ is a constant multiplier we apply to the initialized weights of the model under the Kaiming init. **Right**: the norm of all weights. While all models achieve a 100% training accuracy, models initialized with a larger scale converge to solutions with higher weight norms, which is a sign that the layers are misaligned.

stochastic gradient descent without moment or weight decay are used during training. We use a batch size of 128 and report the best training and test accuracy among all epochs.

We choose $\gamma = \frac{1}{\sqrt{d}}$ and $\eta = 0.05$ for the standard NTK model, $\gamma = \frac{10}{d}$ and learning rate $\eta = 0.05d/100$ for the standard mean-field model, $\gamma = 1$ and $\eta = 0.05d/100$ for the Kaiming$^-$ model, $\gamma = \frac{100}{d}$ and $\eta = 0.05d/100$ for the Kaiming$^+$ model, $\gamma = 1$ and $\eta = 0.05$ for the Xavier$^+$ model, $\gamma = 0.01d$ and $\eta = 0.05(100/d)^2$ for the Xavier$^-$ model. The choice of hyperparameters guarantees that the standard NTK model and the standard mean-field model, the Kaiming$^+$ and Kaiming$^-$ model, and the Xavier$^+$ and Xavier$^-$ model are the same for $d = 100$, respectively.

C.3. Experiments in Section B.3.2

The experiment in Section B.3.2 is similar to that in B.3.1. The only difference is that we fix $d = 2000$ and change the initialization scale. More specifically, we set $\kappa = 10$, $\sigma_u^2 = \kappa^c$, $\sigma_w^2 = \kappa^{\max\{c,0\}}$ and $\gamma = \kappa^{-\min\{0,-c/2\}}$. We also fix $\eta = 0.005$.