New Evidence of the Two-Phase Learning Dynamics of Neural Networks

Zhanpeng Zhou ZZP1012@SJTU.EDU.CN
Shanghai Jiao Tong University, Shanghai, China

Yongyi Yang YONGYI@UMICH.EDU
University of Michigan, Ann Arbor, USA

Mahito Sugiyama MAHITO@NII.AC.JP
National Institute of Informatics, Tokyo, Japan

Junchi Yan YANJUNCHI@SJTU.EDU.CN
Shanghai Jiao Tong University, Shanghai, China

Abstract

Extensive evidence suggests that training dynamics undergo a distinct phase transition, yet our understanding of this transition still lags behind. In this paper, we introduce an interval-wise perspective that compares network states across a time window, revealing two new phenomena that illuminate the two-phase nature of deep learning. i) **The Chaos Effect.** By injecting an imperceptibly small parameter perturbation at various stages, we show that the response of the network to the perturbation exhibits a transition from chaotic to stable, suggesting there is an early critical period where the network is highly sensitive to initial conditions; ii) **The Cone Effect.** Tracking the evolution of the empirical Neural Tangent Kernel (eNTK), we find that after this transition point the model's functional trajectory is confined to a narrow cone-shaped subset: while the kernel continues to change, it gets trapped in a tight angular region. Together, these effects provide a dynamical view of how deep networks transition from sensitive exploration to stable refinement during training.

1. Introduction

Many recent studies have suggested, either implicitly or explicitly, that there is a *phase transition* point during the neural network training, where the model's properties and behaviors undergo substantial shifts before and after this time point. For example, Cohen et al. [4], Damian et al. [5], Wang et al. [18] showed that during training, the network first enters a progressive sharpening phase, and after which, the sharpness remains roughly constant for the rest of the training. Achille et al. [1] identified a critical learning period early in training, during which exposure to low-quality data can cause irreversible damage, while similar exposure later in training can be reversed. Zhou et al. [21] discovered that the late training phase is critical in improving generalization and robustness of neural networks.

Despite abundant evidence for the *two-phase phenomenon* [1, 4, 9, 18–20], a complete characterization and understanding of this phenomenon still lags behind. Moreover, most existing studies adopt a *point-wise* perspective: they primarily focus on examining specific properties of the network at isolated time points. This perspective, while informative, offers only a static snapshot of the model's behavior, and does not capture the temporal dynamics of learning: how a property emerges, evolves, or vanishes as training progresses.

In this paper, to gain a deeper understanding of the two-phase phenomenon, we introduce two new empirical observations that exhibit characteristics of the phase transition. Crucially, these are what we call "interval-wise" phenomena: rather than analyzing a property at specific time points, we compare the model's behavior across two different time points of training. We show that this novel approach reveals patterns that are otherwise invisible to point-wise analysis and offers new insights into the learning dynamics of neural networks. Specifically, we identify and investigate two distinct behaviors: *the Chaos Effect* and *the Cone Effect*.

The Chaos Effect. First, we observe that the learning dynamics of neural networks transition from a chaotic to a non-chaotic regime during training. Specifically, we train two networks that are initialized identically and trained with the same stochastic gradient noise. At a specific time t_0, we apply a small perturbation to the parameters of one model, and then we compare the resulting parameters at a later time t_1. Particularly, we observe an *inflection point* during the training process. We find that when t_0 is in the early stage of training, specifically before the inflection point, even a tiny perturbation leads to a significant divergence from the original training trajectory. This phenomenon indicates a high sensitivity of learned parameters to initial conditions, which is a hallmark of chaotic systems in physics. However, if t_0 is later in training (after the inflection point), the divergence is minimal, suggesting that the system becomes increasingly stable as training progresses.

The Cone Effect. Second, we discover that after the early training phase, the learning dynamics of neural networks remain constrained in a narrow cone in the function space. Specifically, we train a network, and starting from a chosen time point τ, we track the empirical Neural Tangent Kernel (eNTK) at later time steps and measure their deviation from the eNTK at τ. We observe that when τ is sufficiently large, the subsequent eNTKs remain confined within a narrow cone around the eNTKs at time τ. In contrast, if the τ is chosen in the early stages of training, the eNTKs experience chaotic and large unstructured changes over time, and no such confinement is observed.

2. Related Work

As discussed in Section 1, many studies have suggested neural network training undergoes a phase transition in practice. In our view, this body of work can be roughly grouped into the following four categories:

- **Transition of the First-order Quantities.** A recent line of research investigated phase transitions in neural network training using gradient-based metrics. For example, Fort et al. [8] identified a clear branching point in the evolution velocity of the empirical Neural Tangent Kernel (eNTK), transitioning from a fast to a slow regime. Jastrzebski et al. [14] observed that the largest eigenvalue of the gradient covariance matrix increases monotonically in the early phase and then enters a regime of sustained oscillation.

- **Transition of the Second-order Quantities** Extensive works analyzed the Hessian spectrum of neural networks to characterize training phase transitions. For example, Cohen et al. [4], Damian et al. [5], Frankle et al. [10], Wang et al. [18] reported a two-phase pattern in the largest Hessian eigenvalue (the "sharpness") during gradient descent: an initial phase of steady growth in sharpness accompanied by smooth loss reduction, followed by a plateau where sharpness fluctuates around a critical threshold. Ghorbani et al. [11] studied the full Hessian spectrum, clearly revealing a structural phase transition. Gur-Ari et al. [12] showed that the subspace spanned by the top Hessian eigenvectors stabilizes shortly after initialization, indicating a transition in the dominant eigenspace.

• **Transition of the Training Trajectories.** Another line of research focused on the phase transition behavior from the perspective of optimization trajectory and loss landscape geometry. For instance, Fort et al. [8], Frankle et al. [9, 10], Zhou et al. [20] introduced a *spawning method*: a network is initialized and trained for a few epochs, then spawned into two copies that continue training independently under different sources of SGD randomness (e.g., mini-batch order and data augmentation). They showed that when spawning occurs late in training, the two models, despite far in Euclidean distance, remain in the same basin of the loss landscape; whereas early spawning yields models in distinct basins. Singh et al. [17] analyzed the directionality of the optimization path and identified a corresponding phase transition from this directional viewpoint.

• **Transition in Behaviors/Ability.** From a more functional perspective, [19] demonstrate that neural networks learn concepts sequentially based on their signal strength. There exists a critical moment when one concept has been fully learned while another remains unlearned, clearly marking a phase boundary. [1] analyze how models respond to temporary corruptions in training data. They identify a "critical period" early in training during which such temporary corruption can cause permanent damage, while similar interventions later in training are reversible.

3. Preliminary and Methodology

Basic Notations. We focus on a classification task. Denote $[k] = \{1, 2, \cdots, k\}$. Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be the training set of size n, where $\boldsymbol{x}_i \in \mathbb{R}^{d_0}$ represents the i-th input and $y_i \in [c]$ represents the corresponding target. Here, c is the number of classes. Let $f : \mathcal{D} \times \mathbb{R}^p \to \mathbb{R}$ be the NN model, and thus $f(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}$ denotes the output of model f on the input $\boldsymbol{x}$ with parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. Let $\ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}), y_i)$ be the loss at the i-th data point, simplified to $\ell_i(\boldsymbol{\theta})$. The total loss over the dataset $\mathcal{D}$ is then denoted as $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \ell_i(\boldsymbol{\theta})$. We also use $\mathrm{Err}_{\mathcal{D}}(\boldsymbol{\theta})/\mathrm{Acc}_{\mathcal{D}}(\boldsymbol{\theta})$ to denote the classification error/accuracy of the network $f(\boldsymbol{\theta}; \cdot)$ on the training set $\mathcal{D}$.

Parameter Dissimilarity. Following Singh et al. [17], we first introduce the *parameter dissimilarity* to measure the directionality of the optimization process. Specifically, given the training trajectory consisting of a sequence of checkpoints $\{\boldsymbol{\theta}_t\}_{t=0}^T$, we use the pairwise cosine dissimilarity to capture the directional aspect of the trajectory. For any two time points $i, j \in [T]^2$, we define

$$(\boldsymbol{C})_{i,j} := 1 - \cos\langle \mathrm{vec}(\boldsymbol{\theta}_i), \mathrm{vec}(\boldsymbol{\theta}_j)\rangle = 1 - \langle \mathrm{vec}(\boldsymbol{\theta}_i), \mathrm{vec}(\boldsymbol{\theta}_j)\rangle / (\|\mathrm{vec}(\boldsymbol{\theta}_j)\|_2 \, \|\mathrm{vec}(\boldsymbol{\theta}_j)\|_2), \qquad (1)$$

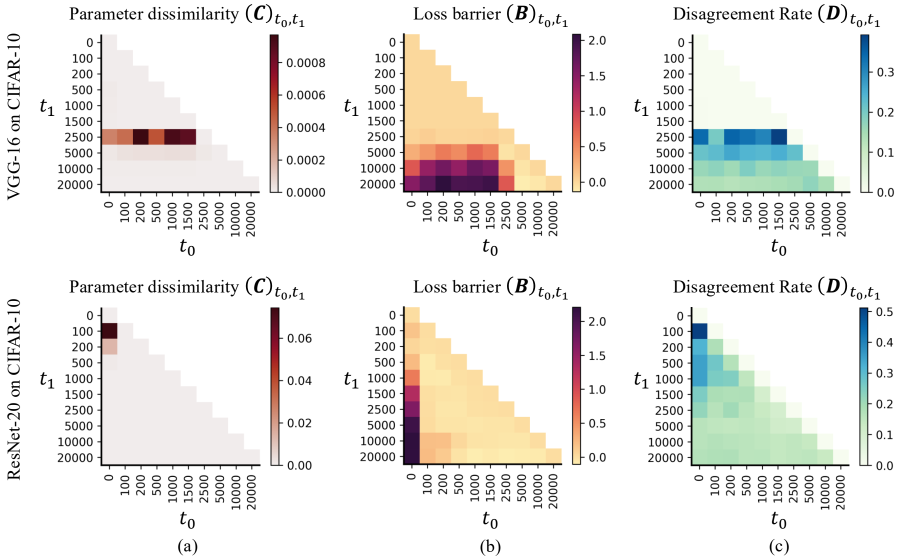
where $\mathrm{vec}(\boldsymbol{\theta})$ denotes the flattened parameters of the network.

Kernel Distance. Similar to Fort et al. [8], we use the *kernel distance* to quantify the evolution of neural networks in the function space. Specifically, we measure the pairwise distance between the *empirical neural tangent kernel* (eNTK) matrix $\boldsymbol{H}(\boldsymbol{\theta})$ at two different time points, namely *kernel distance*. For two time points $i, j \in [T]^2$, we define:

$$(\boldsymbol{S})_{i,j} := 1 - \frac{\langle \boldsymbol{H}(\boldsymbol{\theta}_i), \boldsymbol{H}(\boldsymbol{\theta}_j)\rangle}{\|\boldsymbol{H}(\boldsymbol{\theta})_i\|_F \, \|\boldsymbol{H}(\boldsymbol{\theta}_j)\|_F}. \qquad (2)$$

Loss Barriers. In addition to the directional and functional aspect, we also investigate the geometry of the neural network's loss landscape. In particular, we examine the *loss barriers* [2, 9, 20] between any two points along the training trajectory. For any two points $i, j \in [T]^2$, we define:

$$(\boldsymbol{B})_{i,j} := \max_{\alpha} \mathcal{L}_{\mathcal{D}'}(\alpha\boldsymbol{\theta}_i + (1-\alpha)\boldsymbol{\theta}_j) - \frac{1}{2}(\mathcal{L}_{\mathcal{D}'}(\boldsymbol{\theta}_i) + \mathcal{L}_{\mathcal{D}'}(\boldsymbol{\theta}_j)), \qquad (3)$$



(a)

(b)

(c)

Figure 1: **The sensitivity of learning dynamics to tiny perturbations. (a)** The parameter dissimilarity $(\boldsymbol{C})_{t_0,t_1}$. **(b)** The loss barrier $(\boldsymbol{B})_{t_0,t_1}$. **(c)** The disagreement rate $(\boldsymbol{D})_{t_0,t_1}$. Note that the t_0 and t_1 are presented in iterations, not epochs.

where $\mathcal{D}'$ denotes the unseen test set.

Disagreement Rate. Lastly, we concern about the similarity of the outputs at two points along the optimization trajectory. Specifically, we introduce the *disagreement rate* on the test data for any two points $i, j \in [T]^2$:

$$(\boldsymbol{D})_{i,j} := \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}'}[\mathbf{1}(f(\boldsymbol{x},\boldsymbol{\theta}_i) \neq f(\boldsymbol{x},\boldsymbol{\theta}_j))], \tag{4}$$

where $\mathbf{1}(\cdot)$ is the indicator function and $\mathcal{D}'$ is the test set.

Main Experimental Setup. We train the VGG-16 architecture [16] and the ResNet-20 architecture [13] on the CIFAR-10 dataset. Optimization is done using SGD with momentum (momentum set to 0.9). A weight decay of 1×10^{-4} is applied. The learning rate is initialized at 0.1 and is dropped by 10 times at 80 and 120 epochs. The total number of epochs is 160.

4. The Chaos Effect: Sensitivity of Learning Dynamics to Small Perturbations

In the section, we study the sensitivity of neural network learning dynamics to small perturbations.

Experimental Design. We train two networks with identical initializations and the same stochastic gradient noise. However, at a specific time t_0, we introduce a small perturbation $\boldsymbol{\epsilon}$ to the parameters of one network, such that $\boldsymbol{\theta}'_{t_0} = \boldsymbol{\theta}_{t_0} + \boldsymbol{\epsilon}$. We then compare the resulting models at a later time t_1 ($t_1 > t_0$), with the parameters $\boldsymbol{\theta}'_{t_1}$ and $\boldsymbol{\theta}_{t_1}$ respectively. We consider a tiny perturbation here, where $\|\boldsymbol{\epsilon}\|_0 = 10^{-7}$. We vary the time t_0, t_1 and compare the two resulting models using different metrics.

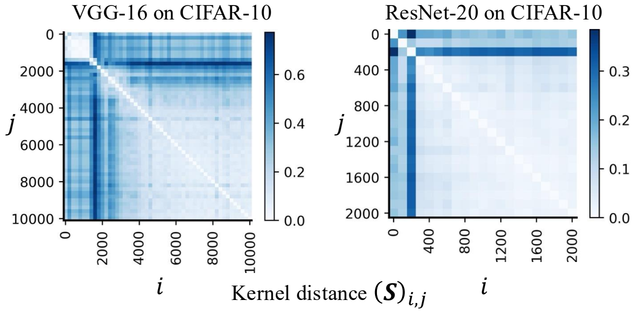
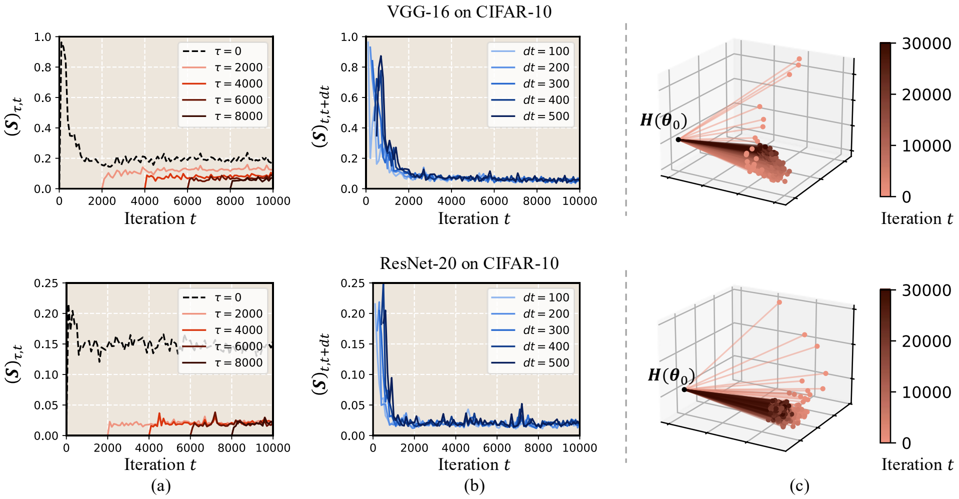


Figure 2: **The kernel distance between every pair of two points at the optimization trajectory $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$.** Our results are reported for both VGG-16 and ResNet-20 on CIFAR-10. Note that the i and j are presented in iterations, not epochs.

Finding I. Optimization trajectory changes its direction at an inflection point. In Figure 1 *(a)*, we present the parameter dissimilarity for any pair of t_0 and t_1 (with $t_1 \geq t_0$). Notably, there exists a specific time t_1 at which the value of $(\boldsymbol{C})_{t_0,t_1}$ remains relatively high for all choices of t_0. Typically, a high value of $(\boldsymbol{C})_{t_0,t_1}$ indicates the directional change along the optimization trajectory. Therefore, the optimization trajectory evidently changes its direction at a fixed point, namely the *inflection point*.

Finding II. Tiny perturbations applied before the inflection point leads to significant loss barriers and disagreement rate. In Figure 1 *(b)*, we also report the loss barrier between each pair of time points t_0, t_1. We observe that even a tiny perturbation ($\|\boldsymbol{\epsilon}\|_0 = 10^{-7}$) applied at a early time point t_0 could result in a substantial loss barrier between the resulting parameters $\boldsymbol{\theta}_{t_1}$ and $\boldsymbol{\theta}'_{t_1}$ in the later stage of training. This indicates that the two solutions likely reside in different, isolated "valleys" of the loss landscape. In Figure 1 *(c)*, we further evaluate the disagreement rate between $\boldsymbol{\theta}_{t_1}$ and $\boldsymbol{\theta}'_{t_1}$. The high disagreement rate validates the functional dissimilarity between $\boldsymbol{\theta}_{t_1}$ and $\boldsymbol{\theta}'_{t_1}$. Together, we show that the learning dynamics pass through a chaotic regime during the early phase of training, where small perturbations might lead to substantial divergence, namely the *chaos effect*.

Conjecture I. The inflection points marks the transition from a chaotic to an non-chaotic regime. Taking VGG-16 on CIFAR-10 as an example, we observe that a significant loss barrier $(\boldsymbol{B})_{t_0,t_1}$ emerges only when $t_0 \leq 2500$ iterations and $t_1 > 2500$ iterations. Similar observations are also noted for the disagreement rate. Recall that the 2500 iteration marks the inflection point for VGG-16 on CIFAR-10. Therefore, we conjecture that *the inflection point serves as a hallmark of the transition from a chaotic to a non-chaotic training regime*. To verify, we compute the kernel distance between every two points along the optimization trajectory. Specifically, we train the neural network and obtain a sequence of checkpoints $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$. Then for any $i, j \in [T]^2$ we compute the kernel distance, i.e., $(\boldsymbol{S})_{i,j}$. In Figure 2, we observe that the eNTKs evolve significantly during the early phase of training, indicating a chaotic regime. Subsequently, the evolution of the eNTKs stabilizes, transitioning to a non-chaotic phase. Notably, the transition point in the evolution of the eNTKs aligns with the inflection points identified in earlier experiments.



(a)

(b)

(c)

Figure 3: **Constrained learning dynamics in the second phase. (a)** The kernel distance $(\boldsymbol{S})_{\tau,t}$ v.s. training iteration t. **(b)** The kernel distance $(\boldsymbol{S})_{t,t+dt}$ vs. training iteration t. **(c)** The visualization of the changes of the eNTK matrices $\boldsymbol{H}(\boldsymbol{\theta}_t)$.

5. The Cone Effect: Constrained Learning Dynamics in the Second Phase

We have seen that the learning dynamics of neural networks undergo a transition from a highly chaotic to a more stable, non-chaotic phase. In this section, we dig deeper into the second "stable" phase. Surprisingly, we find that, contrary to the typical assumption of the *lazy training regime* [3, 6, 7, 15, 22], the neural network continues to evolve. However, this evolution is confined within a narrow, "cone"-like region in function space, namely the *cone effect*.

Beyond the Lazy Regime: The Cone Effect. First, we compute the kernel distance between two adjacent points $\boldsymbol{\theta}(t)$ and $\boldsymbol{\theta}(t + dt)$. In Figure 3 *(b)*, we observe that the kernel distance $(\boldsymbol{S})_{t,t+dt}$ is significant in the early training phase and then drops quickly to a low but non-negligible value. However, surprisingly, we note that in the later training phase, the values of $(\boldsymbol{S})_{t,t+dt}$ are upper-bounded by the same value for different dt. One possible explanation for this phenomenon is that during the second phase, the eNTK matrix evolves in a constrained space.

To validate this, we further measure how the distance between the kernel matrices at the current iterate $\boldsymbol{\theta}_t$ and a referent point $\boldsymbol{\theta}_\tau$ changes during training. In Figure 3 *(a)*, for different referent points τ, the kernel distance $S(\boldsymbol{\theta}_t, \boldsymbol{\theta}_\tau$ first increases and then keeps nearly constant in training. This result suggests that during the second phase, beyond the lazy regime, the model operates in a constrained function space. The visualization in Figure 3 *(c)* further confirms the existence of the cone effect, where a clear "cone" pattern is observed during the evolution of eNTK matrices.

6. Conclusion

In this paper, we introduced an interval-wise perspective on neural network training dynamics. Through this lens, we identified two novel empirical phenomena that characterize a two-phase transition in deep learning: the chaos effect and the cone effect. Together, these findings suggest a transition from an exploratory, unstable phase to a more stable, refinement-oriented phase during training.

References

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. *arXiv preprint arXiv:1711.08856*, 2017.

[2] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *ICLR*. OpenReview.net, 2023.

[3] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019.

[4] Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

[5] Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023.

[6] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 09–15 Jun 2019.

[7] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[8] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

[9] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[10] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020.

[11] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR, 2019.

[12] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[14] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*, 2020.

[15] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] Sidak Pal Singh, Bobby He, Thomas Hofmann, and Bernhard Schölkopf. The directionality of optimization trajectories in neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

[18] Zixuan Wang, Zhouzi Li, and Jian Li. Analyzing sharpness along gd trajectory: Progressive sharpening and edge of stability. *Advances in Neural Information Processing Systems*, 35: 9983–9994, 2022.

[19] Yongyi Yang, Core Francisco Park, Ekdeep Singh Lubana, Maya Okawa, Wei Hu, and Hidenori Tanaka. Swing-by dynamics in concept learning and compositional generalization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going beyond linear mode connectivity: The layerwise linear feature connectivity. *Advances in neural information processing systems*, 36:60853–60877, 2023.

[21] Zhanpeng Zhou, Mingze Wang, Yuchen Mao, Bingrui Li, and Junchi Yan. Sharpness-aware minimization efficiently selects flatter minima late in training. In *The Thirteenth International Conference on Learning Representations*, 2025.

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