HiLD 2024: 2nd Workshop on High-dimensional Learning Dynamics

The Butterfly Effect: Tiny Perturbations Cause Neural Network Training to Diverge

Gül Sena Altıntaş[*] *Mila Quebec AI Institute* GUL-SENA.ALTINTAS@MILA.QUEBEC
Devin Kwok[*] *McGill University and Mila Quebec AI Institute* DEVIN.KWOK@MAIL.MCGILL.CA
David Rolnick *McGill University and Mila Quebec AI Institute* DROLNICK@CS.MCGILL.CA

Abstract

Neural network training begins with a chaotic phase in which the network is sensitive to small perturbations, such as those caused by stochastic gradient descent (SGD). This sensitivity can cause identically initialized networks to diverge both in parameter space and functional similarity. However, the exact degree to which networks are sensitive to perturbation, and the sensitivity of networks as they transition out of the chaotic phase, is unclear. To address this uncertainty, we apply a controlled perturbation at a single point in training time and measure its effect on otherwise identical training trajectories. We find that both the L^2 distance and the loss barrier (increase in loss on the linear path between two networks) for networks trained in this manner increase with perturbation magnitude and how early the perturbation occurs. Finally, we propose a conjecture relating the sensitivity of a network to how easily it is permuted with respect to another network.

1. Introduction

In training a neural network, there are typically a large number of candidate minima that are equivalent in terms of their losses. It is often unclear what differences (if any) exist in the functions corresponding to these minima. For networks trained from the same starting state, functional similarity can be estimated by evaluating the loss (alternatively, error) barrier, which is the maximum increase in loss (error) along a linear path between the network parameters. Formally, the barrier between networks parameterized by θ_A, θ_B is:

$$\sup_{\alpha \in (0,1)} \ell\left(x, y; \alpha\theta_A + (1-\alpha)\theta_B\right) - \alpha\ell\left(x, y; \theta_A\right) - (1-\alpha)\ell\left(x, y; \theta_B\right). \tag{1}$$

where α interpolates between the networks and ℓ is the loss function (0-1 loss for error barriers) for a fixed set of data x, y.[1] Networks with barriers below some noise threshold threshold (2% error in Frankle et al. [6]) are said to exhibit *linear mode connectivity* (LMC). Among other useful properties, LMC is a necessary condition for the loss landscape to be locally convex [14, Definition 3.1]. Thus, barriers can identify when two networks belong to different loss basins [8, 10, 19].

Previous investigations into LMC have found that after a certain point in training, networks consistently converge to the same linearly connected minimum. In particular, Frankle et al. [6] show that stochastic gradient descent (SGD) has noise (e.g. random batch order, data augmentations, and

[*] Equal contribution. Correspondence to devin.kwok@mail.mcgill.ca.
1. As our work is concerned with the shape of the loss landscape on which a network is optimized, we report the cross-entropy loss barrier ($\mathcal{B}_{\text{ce}}$) for training data in all figures of the main text (averaged over three runs).

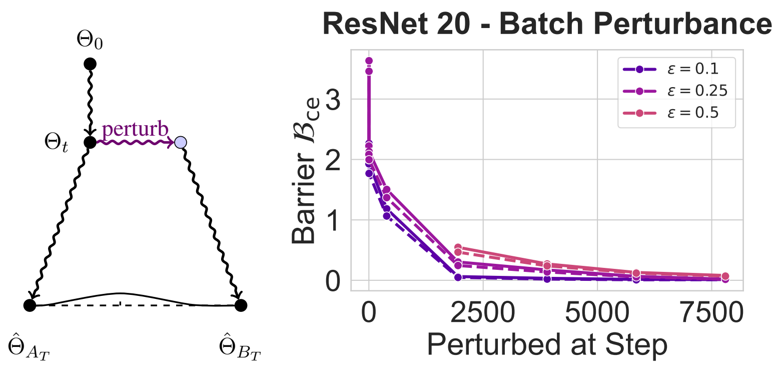

Figure 1: **Left:** illustration of "butterfly effect" experiment. A network θ_0 is trained until time t and perturbed once at t with a perturbation of magnitude ϵ. The original $\hat{\theta}_A$ and perturbed $\hat{\theta}_B$ copies of the network are then trained to convergence at time T, after which the loss barrier between $\hat{\theta}_{A_T}$ and $\hat{\theta}_{B_T}$ is evaluated. **Right:** train (solid) and test (dashed) barriers after training (y-axis) versus perturbation time (x-axis) and magnitude (line color). The earliest perturbations at $\epsilon = 0.1$ are omitted, as training fails to recover the original network's accuracy. See fig. 5 for accuracy barriers.

GPU non-determinism) that causes barriers before a certain time called the *LMC onset* point, but not after that point. Altıntaş et al. [2] find that reducing the variability of early training steps (such as by reducing learning rate, increasing batch size, and using learning rate warmup) can reduce barriers after training. These findings corroborate empirical studies of the neural tangent kernel that suggest training has an initial chaotic phase before networks settle into a consistent loss basin [5].

Although these findings clearly split training into diverging and converging periods, we still do not understand why the chaotic phase of training ceases around the LMC onset point, and how barriers are formed by the process of training. To reach a better understanding of these questions, in this work we ask: how small does a perturbation need to be to avoid causing barriers before the LMC onset, and conversely, how large of a perturbation is needed to cause barriers afterwards? Furthermore, do perturbed training trajectories diverge gradually or suddenly, and how does divergence in parameter space correspond to barriers?

Inspired by studies of twins from genetics, we devise an experiment which investigates how "nature" (the initial state of a neural network) versus "nurture" (the cumulative effect of SGD noise) contribute to the final state of a neural network after training. Our results show that neural network training exhibits the "butterfly effect", wherein a chaotic system is highly sensitive to small changes in its initial state. In particular, we find that:

1. A small perturbation **at a single *early* iteration** is sufficient to induce error barriers between two identically initialized and trained networks (fig. 1).

2. Near initialization, barriers are high for all but the smallest perturbations, while later in training, barriers are low for all but the largest perturbations (fig. 2), with at least 2 orders of magnitude between the scale of these perturbations.

3. Prior works show that small barriers do not necessarily correspond to small L^2 distances between parameters [6, 14, 19]. Nonetheless in our setting, larger barriers are associated with larger L^2 divergences in parameters (fig. 3).

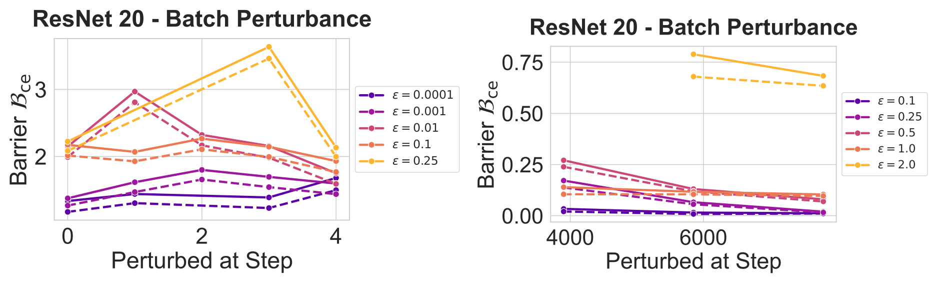

Figure 2: **Left:** inset of train/test (solid/dashed) barriers from fig. 1 for early perturbations, with additional smaller-scale perturbations. **Right:** inset of barriers for late perturbations, with an additional larger-scale perturbation.

4. The above barriers and L^2 distances are inversely correlated with the number of fixed points in a permutation minimizing the L^2 distance between the network parameters after training.

5. Based on other work on neural network permutations [1, 3, 17], we propose a conjecture for networks with sufficiently many permutation symmetries: that barriers arise between an original and a perturbed network when the perturbed network becomes permuted with respect to the original.

2. Single perturbations cause barriers

Previous work focused on the robustness of networks to SGD noise after LMC onset [6], whereas we instead want to examine how sensitive networks are to noise throughout training, particularly *before* LMC onset. We therefore distill the spawning experiment from Frankle et al. [6] to a single step: we train two copies of a network, but instead of perturbing them with independent SGD noise from time t to the end, we apply a single perturbation at t and train with identical SGD noise at all other times (illustrated in fig. 1 left).[2] Appendix A details the training and perturbation methods.

We experiment with two types of perturbations: random batch sampling and additive Gaussian noise. Batch perturbation is equivalent to adding a single extra iteration of SGD training with independent SGD noise scaled by ϵ, so that ϵ functions equivalently to the learning rate (without momentum). Gaussian perturbation adds a Gaussian term with standard deviation proportional to ϵ times the network's random initialization [9]. As larger perturbations can temporarily increase loss, we train for the same amount of time after perturbing the network to ensure convergence. As we find that both types have similar effects, we report results using batch perturbations in the main text. We report Gaussian perturbation results in appendix B and error barriers in appendix C.

Figure 1 shows the effect of batch perturbation time and size on barriers after training. Generally, barriers increase with perturbation scale, but this relationship changes dramatically depending on when the perturbation is applied. Near initialization, adding either a single gradient step (equivalently, Gaussian noise) scaled to as little as 0.01% of the network's gradient (equivalently, random initialization) is sufficient to cause barriers after training (fig. 2, fig. 6). After 5 epochs ($t = 1950$), the same type of perturbation must be at least 2 orders of magnitude larger to cause a barrier. Notably, this shows that sufficiently large perturbations can still cause barriers well after LMC onset.

2. We validate this process by training two unperturbed copies of a network, and confirming they have 0 barrier.

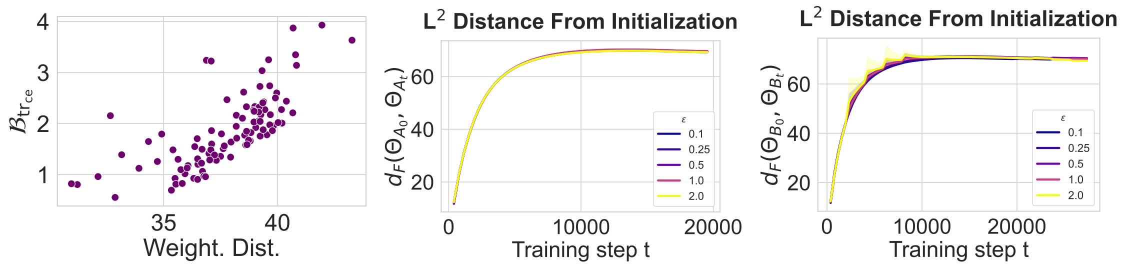

Figure 3: **Left:** barrier (y-axis) after training versus L^2 distance between original and perturbed networks (x-axis). **Center, right:** L^2 distance from initialization (y-axis) over training time (x-axis) for the original (middle) and the perturbed (right) networks.

As the largest perturbations can degrade the trained network's performance, we exclude them from fig. 1 and use learning rate rewinding to evaluate them instead (appendix D).

The Gaussian perturbation method is similar to Frankle et al. [7, section 5.3]. However, Frankle et al. [7] only examined how perturbation reduces training accuracy for pruned networks, and not the degree to which perturbations affect barriers and parameter distances. We also look at a wider range of perturbation sizes and times to better characterize network stability throughout training.

3. Barriers correlate with parameter distance

Prior work has shown that the parameters of LMC networks diverge in terms of L^2 distance [6, 14, 19]. In our setting, we find that although networks with minimal barriers can have a large L^2 distance between their parameters, larger barriers are correlated with even larger L^2 distances after training (fig. 3 left). This divergence is not due to one network travelling further than the other, since both networks are equally distant from initialization after training (fig. 3 middle and right).

4. Barriers inversely correlate with fixed points of weight alignment permutations

Recent work has found that barriers between SGD trained networks can be reduced or even eliminated by taking permutation symmetries into account [1, 3, 17]. Since these prior findings used independently initialized networks, we would expect them to also hold with identically initialized networks in our case. We reverse this question and ask: for networks trained from the same initial state, do barriers arise when they converge to permuted copies of the same loss basin?

Inspired by the proxy model in Entezari et al. [3], we first consider whether the barrier and L^2 distance between perturbed networks in our setting could also be explained by increasingly many permutations, as measured by the proportion of fixed points in a random permutation applied to one of the networks before computing barriers or L^2 distance (fig. 4 left). We find that the fixed points of the random permutations correlate inversely with both L^2 distance and barrier between the original and randomly permuted networks.

We then apply weight alignment [1, 17] to find a permutation that (approximately) minimizes the L^2 distance between the original and perturbed network parameters after training. Figure 4 (center, right) shows that the amount of fixed points in this permutation is also inversely proportional to both barrier size and L^2 distance between the networks before alignment. These observations suggest a possible link between barriers and a permutation of the perturbed network relative to the original.

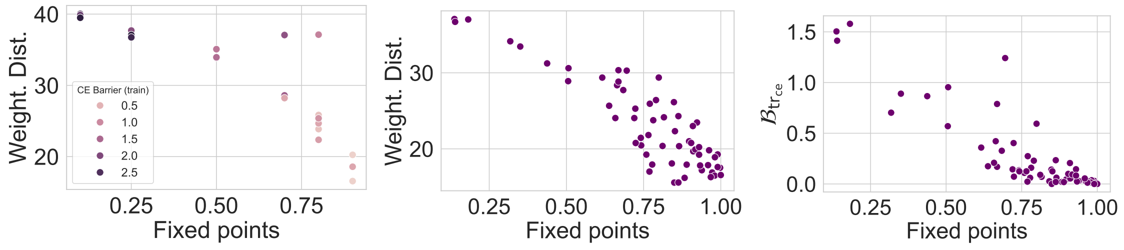

Figure 4: Left: the effect of applying a random permutation with some % of fixed points (x-axis) on a trained network, in terms of barrier (color) and L^2 distance (y-axis) between the original and permuted network. Middle: fixed points versus L^2 distance between original and batch-perturbed network after training. Right: proportion of fixed points (x-axis) in permutation found by weight alignment, versus barrier (y-axis) between original and perturbed network after training.

5. Discussion

Neural network training is generally understood to be sensitive to tiny perturbations [4]. In this work, we present clear empirical evidence for the most extreme version of this phenomena: a tiny perturbation at a single iteration is sufficient for two identically trained networks to diverge (fig. 1). We find that the perturbation needed to induce barriers throughout training is several orders of magnitude smaller near initialization than at later epochs (fig. 2), and also that barriers and distance in parameter space are correlated in our setting (fig. 3).

To explain how these perturbations could lead to barriers, we build on the work of Entezari et al. [3] to propose the following informal conjecture:

Conjecture 1 *For networks trained from the same initial state and with sufficiently many permutation symmetries, with high probability there exists a permutation of one network that eliminates its barrier with the other. Furthermore,* ***barriers arise between networks trained from the same initial state when differences in their training trajectories cause one network to become permuted with respect to the other.***

So far, our preliminary investigations (fig. 4) align with this conjecture.

Since Sharma et al. [17] find that a permutation minimizing barriers between two networks becomes stable to SGD noise shortly after initialization, we speculate that barrier-inducing permutations occur early in training, and could be related to the LMC onset point. Addressing this conjecture will require a method to identify when the permutation connecting two networks changes throughout training. We will formalize this problem in follow up work.

Our investigations have focused on randomly initialized networks, but research also shows that pretrained networks either stay in the same LMC basin when fine-tuned [16], or converge to a few basins representing distinct generalization strategies [11]. It would be useful to know what conditions cause fine-tuning to converge to similar LMC networks versus distinct functions. For instance, applications that combine networks by averaging parameters, such as in continual learning [13], federated learning [12], and model soups [18], perform better on LMC networks by definition. However, greater functional diversity is desirable in applications such as transfer learning [14] and ensembling [16] in order to improve generalization. In follow up work, we will replicate our experiments on pretrained networks to better understand how pretraining affects functional diversity and the barriers between fine-tuned networks.

Acknowledgements

This research was supported by an NSERC Discovery grant, the Canada CIFAR AI Chairs program, and Fonds de recherche du Québec – Nature et Technologies (FRQNT doctoral research award #352816). Computing resources were provided by Mila Quebec AI Institute and the NVIDIA Corporation.

References

[1] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

[2] Gül Sena Altıntaş, Gregor Bachmann, Lorenzo Noci, and Thomas Hofmann. Disentangling linear mode connectivity. In *UniReps: the First Workshop on Unifying Representations in Neural Models*, 2023. URL https://openreview.net/forum?id=PbvPwiySXz.

[3] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

[4] Emilio Ferrara. The butterfly effect in artificial intelligence systems: Implications for ai bias and fairness. *Machine Learning with Applications*, 15:100525, 2024. ISSN 2666-8270. doi: https://doi.org/10.1016/j.mlwa.2024.100525. URL https://www.sciencedirect.com/science/article/pii/S266682702400001X.

[5] 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. URL https://proceedings.neurips.cc/paper/2020/hash/405075699f065e43581f27d67bb68478-Abstract.html.

[6] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3259–3269. PMLR, 2020. URL https://proceedings.mlr.press/v119/frankle20a.html.

[7] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkl1iRNFwS.

[8] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively characterizing neural network optimization problems, 2015. URL https://arxiv.org/abs/1412.6544.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December

2015. URL https://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html.

[10] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BJYwwY9ll.

[11] Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=hY6M0JHl3uL.

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

[13] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear Mode Connectivity in Multitask and Continual Learning, October 2020. URL http://arxiv.org/abs/2010.04495. arXiv:2010.04495 [cs].

[14] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv:2008.11687 [cs, stat]*, 2020. URL http://arxiv.org/abs/2008.11687. arXiv: 2008.11687.

[15] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gSj0NKvB.

[16] Ildus Sadrtdinov, Dmitrii Pozdeev, Dmitry P Vetrov, and Ekaterina Lobacheva. To stay or not to stay in the pre-train basin: Insights on ensembling in transfer learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 15936–15964, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/336572db3e99930814d6b328d4220cb6-Paper-Conference.pdf.

[17] Ekansh Sharma, Devin Kwok, Tom Denton, Daniel M. Roy, David Rolnick, and Gintare Karolina Dziugaite. Simultaneous linear connectivity of neural networks modulo permutation, 2024. URL https://arxiv.org/abs/2404.06498.

[18] Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 11217–11227. PMLR, 2021. URL https://proceedings.mlr.press/v139/wortsman21a.html.

[19] David Yunis, Kumar Kshitij Patel, Pedro Henrique Pamplona Savarese, Gal Vardi, Jonathan Frankle, Matthew Walter, Karen Livescu, and Michael Maire. On convexity and linear mode connectivity in neural networks. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022. URL https://openreview.net/forum?id=TZQ3PKL3fPr.

Appendix A. Experimental details

Training details. In all experiments we train a ResNet-20 [9] with a width of 32 on CiFAR-10 on a single GPU. We replace batch normalization layers with layer normalization. The parameters are initialized according to Kaiming/He normal initialization. The model is trained using stochastic gradient descent (SGD) with a momentum of 0.9 and a triangular learning rate schedule (linear warmup and decay), with 1 epoch of warmup to a peak learning rate of 0.1. We train the model with a batch size of 128 for 50 epochs, i.e. for a total of 19,500 updates. We employ random horizontal flips, random translation and cutout augmentations.

Perturbation Details. The batch perturbation procedure operates as follows: a random batch of examples is drawn and used to conduct a single forward/backward pass of SGD on the perturbed network at perturbation time. The gradient from this pass is saved, and the perturbed network is reset to the original network at the perturbation point. This gradient is then scaled by multiplying by ϵ and added to the perturbed network's parameters:

$$\hat{\theta}_{B_t} = \theta_t + \epsilon \cdot \delta_{\text{batch}}, \qquad \delta_{\text{batch}} = \nabla \ell(x, y; \theta_t), \qquad x, y \sim \mathcal{D}$$

where $\hat{\theta}_{B_t}$ and θ_t are the perturbed and original networks respectively at time t, ϵ is the perturbation scale, $\nabla \ell$ the gradient of the loss function with respect to θ_t, and x, y are the randomly drawn batch from the dataset $\mathcal{D}$.

The Gaussian perturbation procedure operates as follows: a random sample is drawn from the network's initial distribution. This is multiplied by ϵ and added to the perturbed network's parameters at perturbation time.

$$\hat{\theta}_{B_t} = \theta_t + \epsilon \cdot \delta, \qquad \delta_{ij}^{(l)} \sim \mathcal{N}\left(0, \frac{2}{n_{l-1}}\right)$$

where $\delta_{ij}^{(l)}$ are parameters from layer l from independently drawn from the Kaiming/He initialization, which is a Gaussian with standard deviation $\frac{2}{n_{l-1}}$, where n_{l-1} is the number of neurons in the previous layer [9].

To ensure that the perturbed network has sufficient training time to converge even when perturbed late in training, the perturbed network is always trained for exactly 50 epochs (19,500 updates) after the perturbation time. Specifically, the original network is trained for an additional $T-t$ steps after perturbation time t for a total training time of $T = 19500$, whereas the perturbed network is trained for T more steps after perturbation for a total of $T + t$ updates.

Appendix B. Perturbations with additive Gaussian noise

We repeat our experiments using perturbations defined by adding Gaussian noise proportional to the random initialization of the network, instead of randomly sampled batch gradient updates. The magnitude of the perturbation is indicated by the scale ϵ by which the perturbation is multiplied, with $\epsilon = 1$ corresponding to the distribution of the network's parameters at random initialization. Figures 5 to 8 correspond to figs. 1 to 4 from the main text respectively.

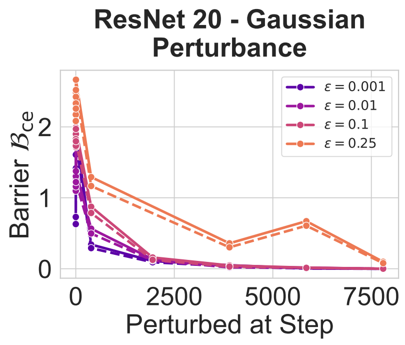

Figure 5: Identical plot to fig. 1, but with Gaussian perturbation. Barriers at the end of training (y-axis) versus perturbation time (x-axis) and magnitude (line color) for Gaussian perturbations. Means over three runs are plotted.

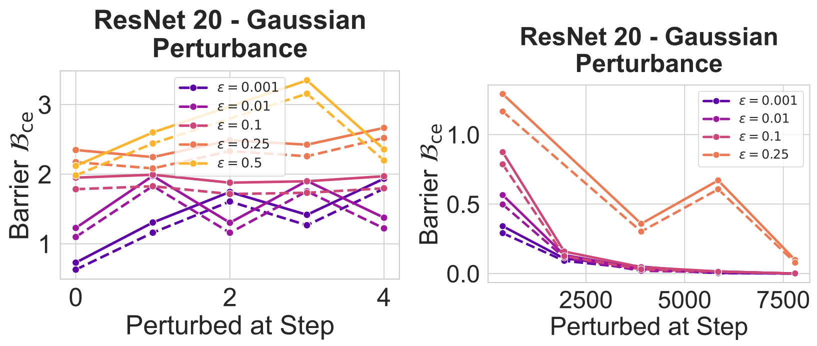

Figure 6: Identical plot to fig. 2, but with Gaussian perturbation. **Left:** inset of barriers for early training perturbations, with additional smaller perturbations. **Right:** inset of barriers for late training perturbations, with additional larger perturbations.

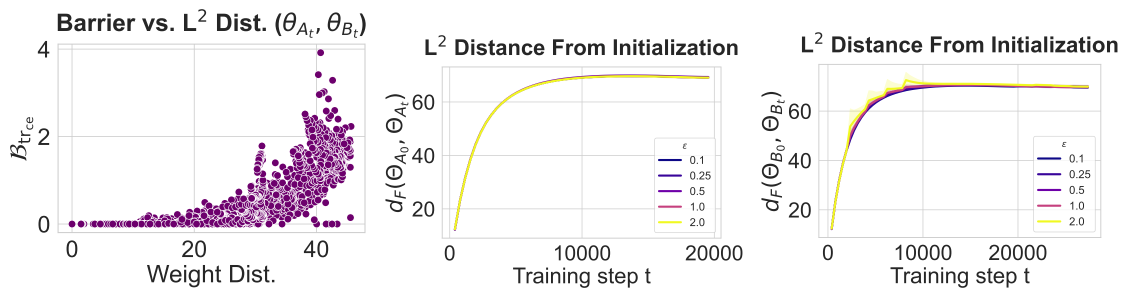

Figure 7: Identical plot to fig. 3, but with Gaussian perturbation. **Left:** L^2 distance between original and perturbed network (x-axis), versus barrier (y-axis) after training. **Center, right:** L^2 distance from initialization (y-axis) over training time (x-axis) for original (middle) and perturbed (right) networks.

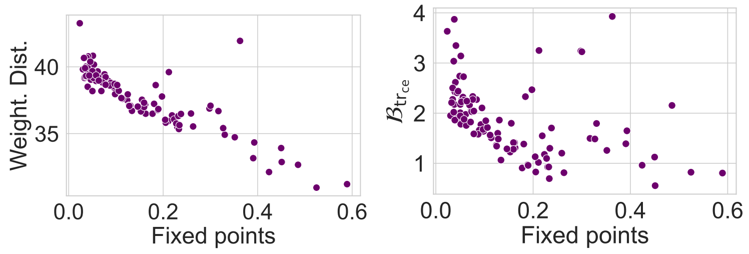

Figure 8: Identical plot to fig. 4 (middle and right), but with Gaussian perturbation. **Left:** ratio of fixed points (x-axis) of permutation found by weight alignment versus L^2 distance between original and perturbed network after training. **Right:** ratio of fixed points (x-axis) of permutation found by weight alignment versus barrier between original and perturbed network after training.

Appendix C. Error barriers

Here, we report the same results in fig. 1 but with error barrier (maximum accuracy decrease over the linear path between two networks) on the y-axis for figs. 9 and 10. We also report the error barriers for the networks trained with Gaussian perturbations in figs. 11 and 12. Again, means over three runs are plotted.

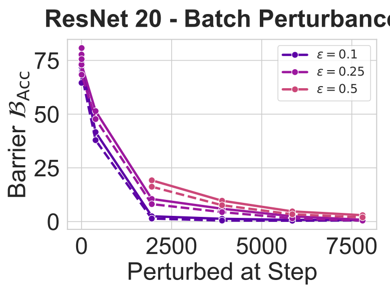

Figure 9: Identical plot to fig. 1, but plotting error barrier as a percentage, i.e. $100(1 - \text{accuracy})$. Barriers at the end of training (y-axis) versus perturbation time (x-axis) and magnitude (line color) for batch-gradient perturbations.

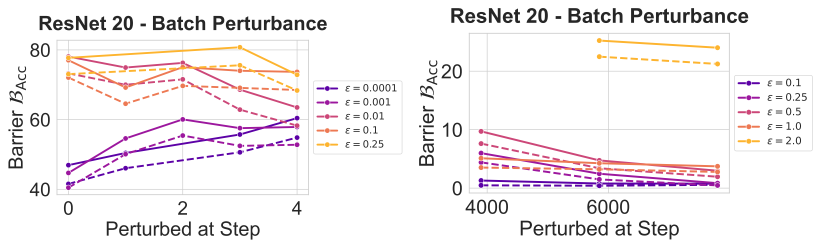

Figure 10: Identical plot to fig. 2, but with error barriers. **Left:** inset of barriers for early training perturbations, with additional smaller perturbations. **Right:** inset of barriers for late training perturbations, with additional larger perturbations.

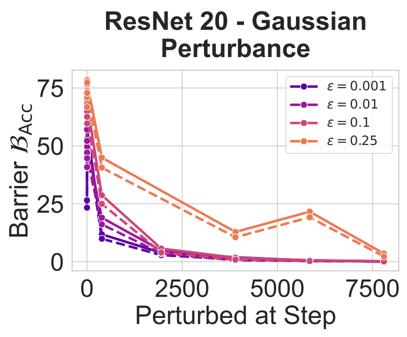

Figure 11: Identical plot to fig. 5 with Gaussian perturbation, but with error barriers. Error barriers at the end of training (y-axis) versus perturbation time (x-axis) and magnitude (line color) for batch-gradient perturbations. Means over three runs are plotted.

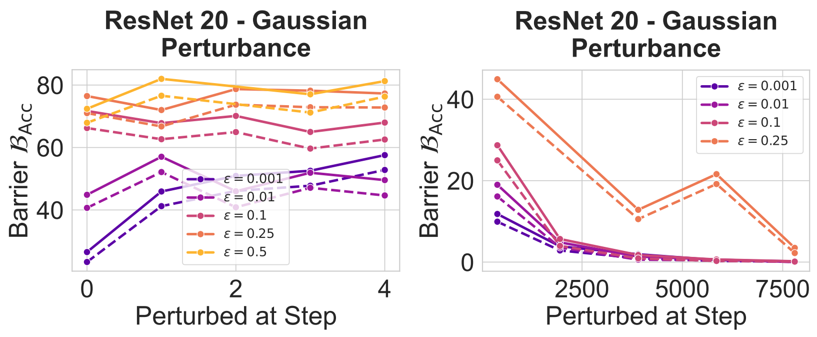

Figure 12: Identical plot to fig. 6 with Gaussian perturbation, but with error barriers. Left: inset of barriers for early training perturbations, with additional smaller perturbations. Right: inset of barriers for late training perturbations, with additional larger perturbations.

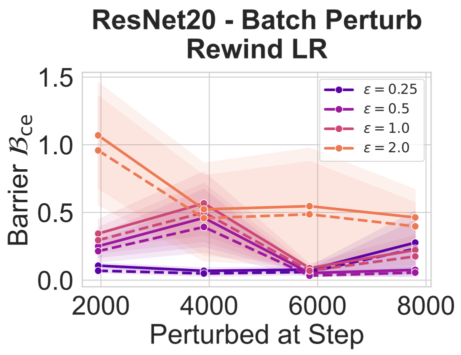

Figure 13: Identical plot to fig. 2, but with the learning rate scheduler reset to initialization after perturbation. Barriers at the end of training (y-axis) versus perturbation time (x-axis) and magnitude (line color).

Appendix D. Learning rate rewinding

We repeat our experiments with batch perturbation, but here we additionally reset the learning rate scheduler of the perturbed model to its state at initialization—a process called *learning rate rewinding* [15]. Rewinding the learning rate allows us to experiment with larger perturbation sizes that would otherwise compromise the performance of the perturbed model. In fig. 13, we observe that the earlier and larger perturbations result in higher barriers, supporting previously observed trends (fig. 2 right), albeit with a larger variation in barriers. This is expected since rewinding the learning rate results in larger step sizes which amplify SGD noise, especially when the learning rate becomes larger after rewinding when compared with the original learning rate schedule. Nonetheless, the barriers later in training remain significantly lower than those near initialization (fig. 2, left).