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

Linear Weight Interpolation Leads to Transient Performance Gains

Gaurav Iyer GAURAV.IYER@MILA.QUEBEC
McGill University, Mila - Quebec AI Institute

Gintare Karolina Dziugaite GKDZ@GOOGLE.COM
Google Deepmind, McGill University, Mila - Quebec AI Institute

David Rolnick DROLNICK@MILA.QUEBEC
McGill University, Mila - Quebec AI Institute

Abstract

We train copies of a neural network on different sets of SGD noise and find that linearly interpolating their weights can, remarkably, produce networks that perform significantly better than the original networks. However, such interpolated networks consistently end up in unfavorable regions of the optimization landscape: with further training, their performance fails to improve or degrades, effectively undoing the performance gained from the interpolation. We identify two quantities that impact an interpolated network's performance and relate our observations to linear mode connectivity. Finally, we investigate this phenomenon from the lens of example importance and find that performance improves and degrades almost exclusively on the harder subsets of the training data, while performance is stable on the easier subsets. Our work represents a step towards a better understanding of neural network loss landscapes and weight interpolation.

1. Introduction

Linear interpolations of neural network weights are of considerable interest in modern deep learning [17, 21, 26]. They have aided our understanding of training dynamics and loss landscapes [7, 23, 25], and can improve performance at convergence [13, 18, 28]. Much of the prior work on this topic has focused on linear interpolations of network weights during late-stage training or at convergence [28], or interpolating SGD iterates of a network along a single training trajectory [29]. However, our understanding of the properties of such networks and their optimization remains limited in more general settings. The observations and investigations presented here attempt to address this gap and improve our understanding of how linear interpolations evolve throughout the training process.

We train copies of networks on different sets of SGD noise on standard vision tasks. More specifically, we consider a network initialization that is trained for a small number of iterations k, after which its copies A and B are trained on different sets of SGD noise for s epochs. Our main findings and contributions are summarized as follows:

- We show that the performance of the network derived by interpolating A and B (referred to as the ***interpolant*** henceforth) varies significantly throughout the optimization process. Specifically, given k, the interpolant can display better accuracy than A or B on the training and test set, if the value of s is small enough. Later in training (i.e. as s becomes larger), interpolant performance becomes significantly worse.

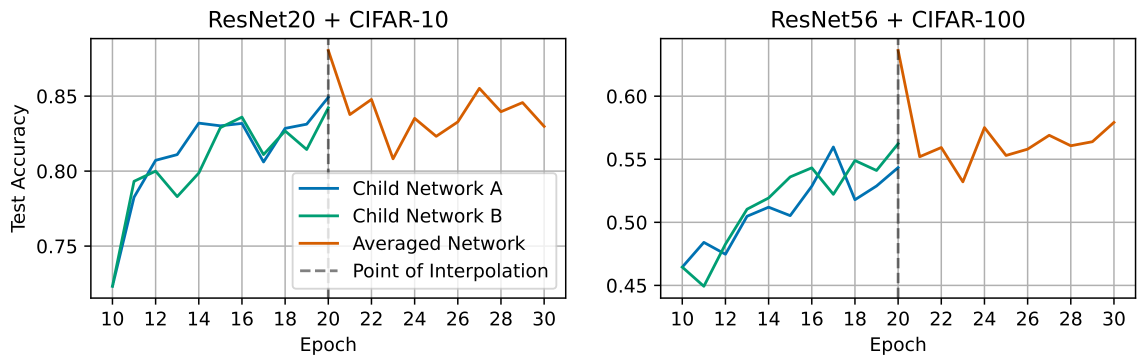

Figure 1: A network is trained for 10 epochs, after which, two copies are made and trained on different sets of SGD noise, and weight averaged. The resulting network is trained further. We find that test accuracy shoots up upon interpolation, but then precipitously drops.

- We attempt to train interpolants derived early in training and find that the improvement in performance is seemingly transient – while they display better immediate performance, they either fail to improve or degrade over the next few epochs of training. Empirically, standard learning rate schedule adjustments do not enable speed-ups in optimization. We hypothesize that first-order optimization methods may be unsuitable for training interpolants and maintaining the accuracy boost.

- We analyze this phenomenon through the lens of example importance for good generalization, and find that "important" examples (those with higher EL2N scores as defined in Paul et al. [22]) contribute significantly to the boost *and* subsequent drop in performance.

2. Interpolating Weights Early in Training

Understanding the effect of linear interpolation early in training can improve our understanding of linear connectivity since it is well-known that the early, noisy stage of training can heavily impact training dynamics and performance at later stages [6, 9, 10, 14, 24]. Our analysis helps tease apart the properties and structure of neural network loss landscapes at different stages during training.

Similar to the experimental setup in Frankle et al. [7], we start with a network that is randomly initialized or trained for k steps. Then, two copies of this network, A and B, are trained on different sets of SGD noise for s epochs. Experimental details for the training setup can be found in Appendix B. We will show the results of our experiment on CIFAR-10 [15] with ResNet20 networks [11] here (specifically Fig. 2 in this context) – additional results on different datasets and network architectures can be found in Appendix D and onward.

We observe that interpolant performance depends on both k and s. When $k = 0$, interpolants perform poorly. However, when the value of k is increased slightly, we make two observations: (a) the interpolants slightly outperform networks A and B early in training, and (b) the rate at which interpolant performance degrades becomes smaller. As k scales, both these observations become more dominant – interpolants outperform the original networks for a longer portion of the training period, and their performance degrades more slowly. Note that the interpolants are not trained further – their performance is simply noted while networks A and B continue training on separate SGD trajectories.

Once k is sufficiently large, interpolant performance improves by more or less the same amount regardless of the exact value of s. This is important from a practical perspective – we can potentially

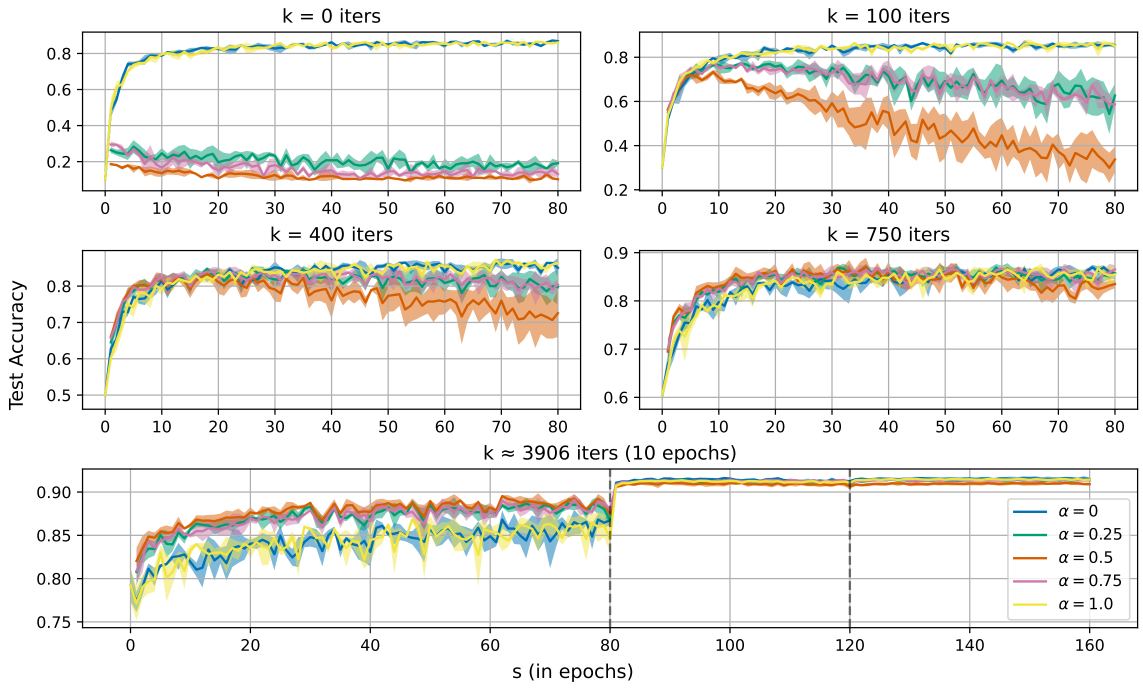

Figure 2: Networks trained for k steps before being cloned into child networks, after which they are trained on different SGD trajectories for s epochs. The vertical dotted black lines represent learning rate drops by a factor of 0.1, and α is the interpolation timestep. Note that the interpolants are not trained further after interpolation – their performance is only recorded while child networks are separately trained.

improve the performance of a network by simply training two copies on different sets of SGD noise for a small number of steps and interpolating their weights.

Observation (b) is partly captured by Frankle et al. [7] – they show that instability to SGD noise i.e. the performance gap between networks A and B and their interpolants at convergence gradually decreases as k increases. When k becomes sufficiently large, the interpolated weights perform better than or approximately as well as networks A and B at all stages of training, making the network "stable to SGD noise". However, we believe that the picture is still far from clear and does not explain observation (a) at all.

3. Training Linear Interpolants

Our findings so far lead to a natural question: if linearly interpolating the weights of networks leads to better performance, can we efficiently leverage this to speed up optimization?

We address this by looking at the simplest case – two copies of a network are made after k epochs, trained on different sets of SGD noise, and interpolated. The resulting interpolant is then trained further. Note from the previous section, that improvement/degradation in performance is most significant for $\alpha = 0.5$ – we will use this setting in our further experiments. Our observations will generally hold for other reasonable values of α.

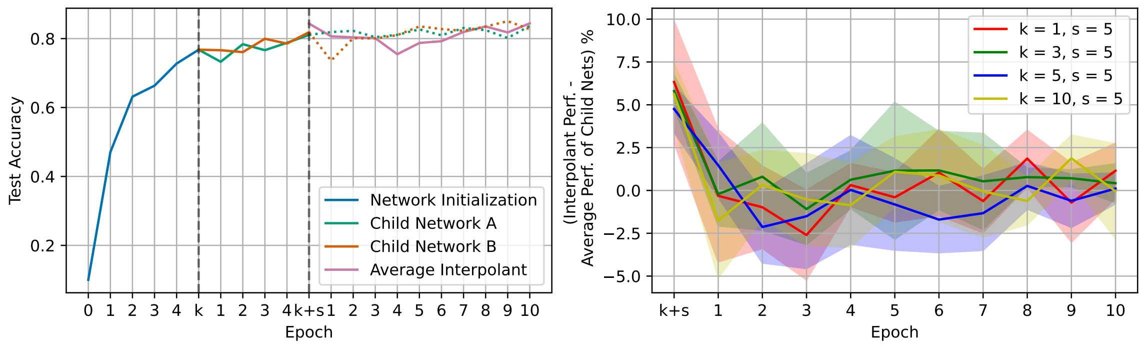

Figure 3: (a) Test performance of child networks and interpolant before and after weight averaging, for $k = 5$ and $s = 5$ epochs. We observe a rise in test accuracy followed by an immediate drop. (b) Difference between interpolant and child network performance, for various values of k. We consistently observe that the interpolant performs significantly better than the child networks at the point of interpolation, but there is effectively no difference after 1-2 epochs. See Fig. 5 for variations.

We train ResNet-20 networks on CIFAR-10 for different values of k and s (see Fig. 3 and Fig. 5). We find that while interpolating weights improves performance, it stagnates or degrades immediately over the next few epochs of further training. Simply dropping the learning rate after interpolation removes this observed drop in performance, but lowers the performance at convergence, which may be undesirable (see Fig. 6). As a compromise, we try warming up the learning rate after interpolation and find that this does not change the original behavior (see Fig. 7).

This suggests that networks obtained by weight interpolation consistently end up in locally "better" regions of the loss landscape, but are not naturally amenable to further training through SGD – the drop in performance heavily implies that training is stalled as the interpolated network escapes its present position in the loss landscape. Our results raise two key questions:

- Why does linear interpolation of network weights trained from the same initialization on different sets of SGD noise lead to networks that perform consistently and significantly better? How does this depend on quantities k and s?
- Why are interpolants derived in this manner not naturally amenable to further training through SGD? What are the properties of interpolants and their immediate positions in weight space or the loss landscape that cause this, and if and how can this be prevented or mitigated?

4. Example Importance

We now consider the following questions: Are there specific examples in the training set that are impacted more heavily than others by weight interpolation? If so, is it the same set of examples that contribute significantly to the improvement and subsequent drop in performance?

The Error L2-Norm (EL2N) score was introduced by Paul et al. [22] as an example importance metric. The authors demonstrate that the higher-scoring examples are *more difficult* to learn but are also more *important for generalization* – removing these difficult examples from the training set had the biggest effect on the final generalization of the model.

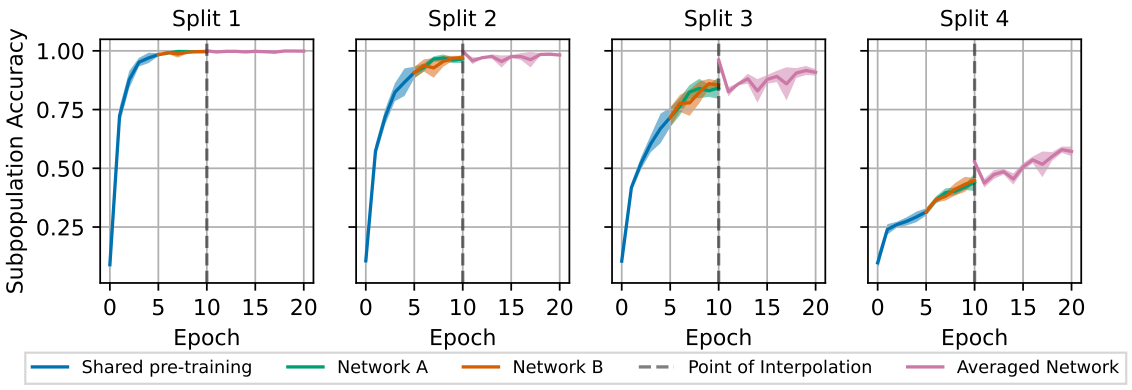

Figure 4: Network performance on training set splits based on EL2N score. Split 1 contains the least important examples, and Split 4 contains the most important ones. The variation in performance due to interpolation is most noticeable in Split 3 and Split 4. As noted in prior work [22], the network is likely to be unstable to SGD noise with respect to these splits early in training, supporting our observations.

Maintaining the previous experiment setup, we track the performance of the child networks and the averaged network across 4 equal splits of the training set (see Fig. 4). The data is split according to the EL2N scores of the training examples computed at epoch 10 of training with Split 1 and Split 4 containing examples with the lowest and highest EL2N scores respectively.

As expected and shown by previous work, more important examples (as defined above) are indeed learned later in training. We also observe that it is primarily on Splits 3 and 4 that the network's performance improves and subsequently drops. While the former may seem obvious (these are the only splits where there is space for performance to improve), the latter is not – linearly interpolating the weights causes a drop in performance almost exclusively on the harder examples in a dataset. Furthermore, the network maintains stellar performance on the easier examples when trained further.

5. Conclusion and Discussion

We have shown that when copies of a neural network are trained on different sets of SGD noise, the network resulting from a linear interpolation of their weights can perform better than the trained copies. However, when further trained, their performance fails to improve or degrades. We analyzed this phenomenon through the lens of example importance and showed that the observed improvement and degradation in performance occur specifically in data subsets that are more important for generalization.

The strategies we explore to mitigate the drop in performance are based on previously used optimization techniques; due to the empirical nature of our study, we cannot conclude whether alternatives exist that could robustly optimize the interpolants. Furthermore, current observations may only apply to the standard vision tasks and not other domains.

Overall, our work provides insight into the behavior of network weight interpolations during the early stages of training and at the same time, raises new questions to address. We believe that this will help us better understand neural network loss landscapes and weight interpolation in the future.

6. Acknowledgments

This research was supported by an NSERC Discovery grant and the Canada CIFAR AI Chairs program. The authors acknowledge material support from NVIDIA and Intel in the form of computational resources and are grateful for technical support from the Mila IDT team in maintaining the Mila Compute Cluster. The authors would also like to thank Devin Kwok for helpful feedback and discussions in the early stages of this work, and Amr Khalifa for feedback on a draft.

References

[1] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging Models Modulo Permutation Symmetries. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

[2] Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep Learning Through the Lens of Example Difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889, 2021.

[3] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. CINIC-10 is not ImageNet or CIFAR-10, 2018.

[4] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially No Barriers in Neural Network Energy Landscape. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/draxler18a.html.

[5] 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.

[6] 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.

[7] 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.

[8] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *Advances in Neural Information Processing Systems*, 31, 2018.

[9] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Edward Dahl, Zachary Nado, and Orhan Firat. A Loss Curvature Perspective on Training Instabilities of Deep Learning Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=OcKMT-36vUs.

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing Models with Task Arithmetic. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

[13] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging Weights Leads To Wider Optima and Better Generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[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. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

[15] Alex Krizhevsky. *Learning Multiple Layers of Features From Tiny Images*. PhD thesis, University of Toronto, 2009.

[16] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-Train-Merge: Embarrassingly Parallel Training of Expert Language Models. *arXiv preprint arXiv:2208.03306*, 2022.

[17] Tao Li, Zhehao Huang, Qinghua Tao, Yingwen Wu, and Xiaolin Huang. Trainable Weight Averaging: Efficient Training by Optimizing Historical Solutions. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8wbnpOJY-f.

[18] Michael S Matena and Colin A Raffel. Merging Models with Fisher-Weighted Averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[19] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In

Aarti Singh and Jerry Zhu, editors, *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.

[20] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear Mode Connectivity in Multitask and Continual Learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Fmg_fQYUejf.

[21] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What Is Being Transferred In Transfer Learning? *Advances in neural information processing systems*, 33:512–523, 2020.

[22] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep Learning on a Data Diet: Finding Important Examples Early in Training. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Uj7pF-D-YvT.

[23] Mansheej Paul, Feng Chen, Brett W Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? *arXiv preprint arXiv:2210.03044*, 2022.

[24] Mansheej Paul, Brett Larsen, Surya Ganguli, Jonathan Frankle, and Gintare Karolina Dziugaite. Lottery tickets on a data diet: Finding initializations with sparse trainable networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 18916–18928. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/77dd8e90fe833eba5fae86cf017d7a56-Paper-Conference.pdf.

[25] Ekansh Sharma, Devin Kwok, Tom Denton, Daniel M. Roy, David Rolnick, and Gintare Karolina Dziugaite. Simultaneous Linear Connectivity of Neural Networks Modulo Permutation, 2024.

[26] Sidak Pal Singh and Martin Jaggi. Model Fusion via Optimal Transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[27] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An Empirical Study of Example Forgetting During Deep Neural Network Learning. *arXiv preprint arXiv:1812.05159*, 2018.

[28] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[29] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead Optimizer: k Steps Forward, 1 Step Back. *Advances in Neural Information Processing Systems*, 32, 2019.

[30] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going Beyond Linear Mode Connectivity: The Layerwise Linear Feature Connectivity. *arXiv preprint arXiv:2307.08286*, 2023.

Appendix A. Related Work

Network weight interpolation has attracted significant interest in recent work. Izmailov et al. [13] introduces Stochastic Weight Averaging, where averaging SGD iterates along the same trajectory lead to better generalization. In a similar vein, Zhang et al. [29] introduces the Lookahead optimizer, which interpolates a set of "fast weights" to update the actual weights of the network, leading to better training stability and generalization. Wortsman et al. [28] show that networks fine-tuned with different hyperparameter configurations often lie in the same loss basin, and that averaging them can lead to better robustness and performance. Ilharco et al. [12] introduce the notion of "task vectors", which capture task-specific directions in weight space. Furthermore, they can be arithmetically combined with fine-tuned network weights and with each other to improve and/or degrade performance on specific tasks. Several other works support these observations [16, 18, 19]. In contrast to these settings, our work focuses on interpolants obtained in the earlier stages of training.

Our work is largely based on the framework provided by Frankle et al. [7], which introduced linear mode connectivity – networks trained from the same initialization on different sets of SGD noise (after some short period of shared training) converge to the same linearly connected minimum. We build upon their work by investigating interpolants earlier in training, instead of at convergence. Mirzadeh et al. [20] observe that networks starting with the same initialization are connected by linear, low-loss paths in the context of continual and multitask learning. Zhou et al. [30] show that this notion of linear mode connectivity extends to layer-wise feature maps as well. Permutation symmetries have been shown to align networks so that they become linearly mode connected [1, 5, 25]. Garipov et al. [8] and Draxler et al. [4] show that networks trained from different initializations can be connected by non-linear low-loss paths.

Paul et al. [22] propose the GraNd and EL2N scores to identify important examples early in training across different network architectures and training configurations. We make use of the EL2N score to analyze observations made in this work and relate it to example importance. Toneva et al. [27] shows that deep networks learn "easy" data earlier in training and rarely forget this subset. Furthermore, these examples do not contribute significantly to final generalization performance. Our experiments support these findings – we find that interpolated network performance is most variable on examples that are learned late in training. Baldock et al. [2] introduces the notion of effective prediction depth to empirically measure example importance and correlate it to the accuracy and speed of learning a given example.

Appendix B. Experimental Setup

For all experiments, networks were trained with a batch size of 128 for a total of 160 epochs. A stepwise learning rate schedule was employed with an initial learning rate of 0.1 – the learning rate is reduced by a factor of 10 at epoch 80 and epoch 120. When linear warmup is employed over t iterations, the learning rate is $\frac{i}{t} \times l$ at iteration i where l is the initial learning rate. All datasets were also augmented with random crops to a size of 32×32 pixels and horizontal image flips with a probability of 0.5. Error bars provided in figures are over 3 network initializations.

Appendix C. Additional Results for ResNet20 + CIFAR-10

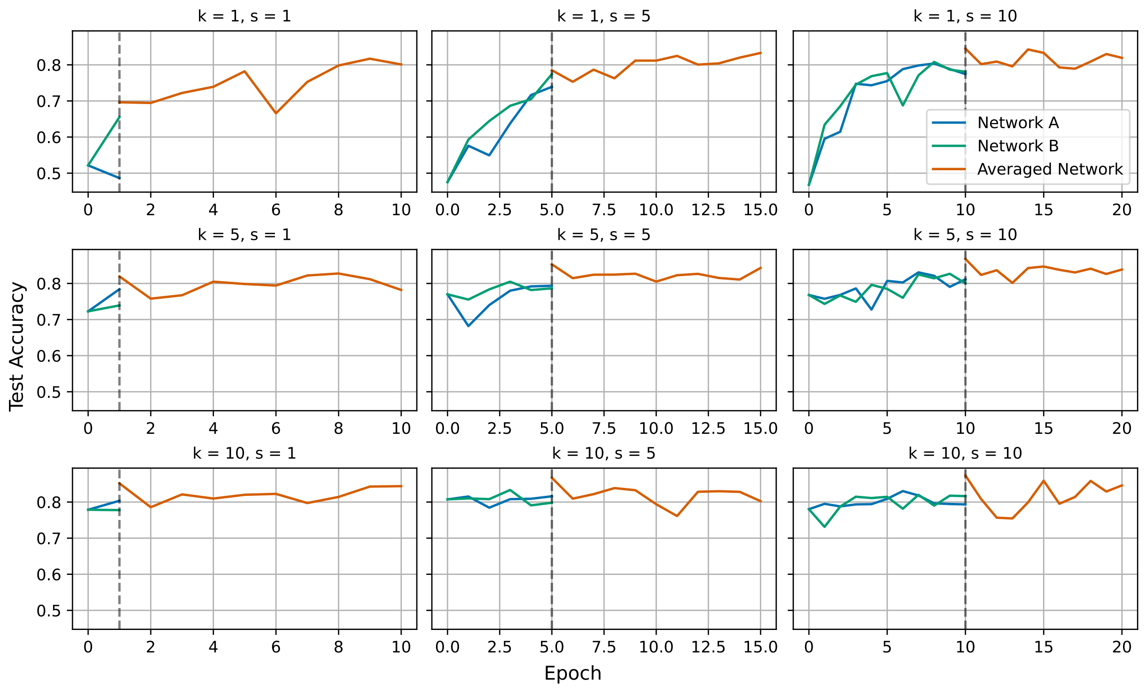

Figure 5: Zoomed-in view of test performance of child networks before interpolation and the weight-averaged network at, and after interpolation, for a variety of k and s values in epochs. We consistently observe a rise in test accuracy followed by an immediate pause in improvement or drop over the next few epochs. Interestingly, the magnitude of improvement seems to be independent of the exact values of k and s.

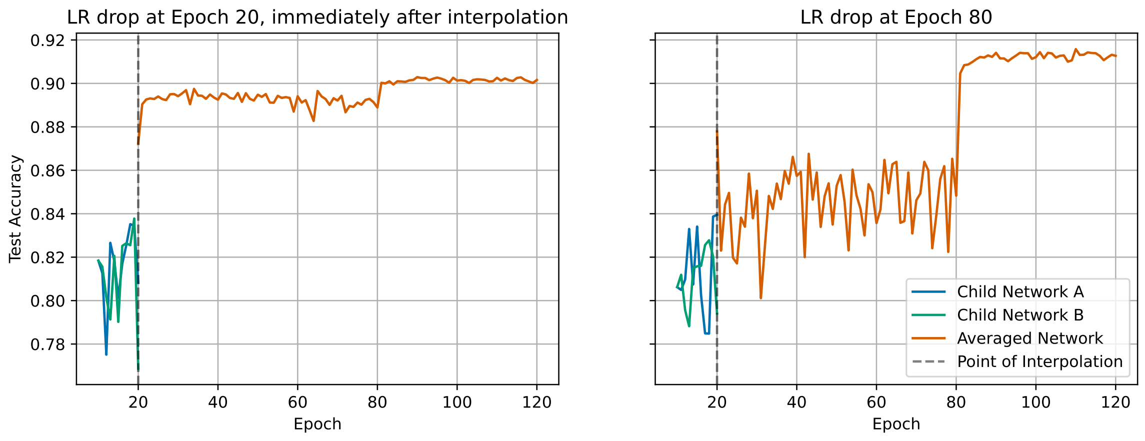

Figure 6: Dropping learning rate early in training removes the performance degradation observed in Fig. 3 but affects final performance significantly.

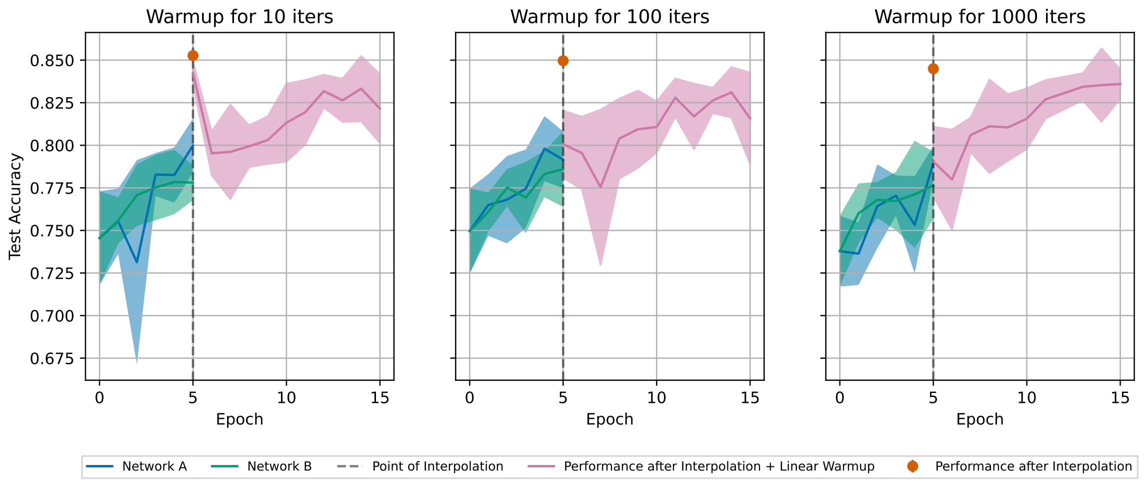

Figure 7: Zoomed-in view of test performance of child networks before interpolation and the weight-averaged network at, and after interpolation, for a variety of $k = 5$ and $s = 5$ epochs – note that the plot begins *after* the first k epochs. Test performance of the averaged network continues dropping even when linear warmup is employed after interpolation. The learning rate is warmed up over t iterations, where the learning rate is $\frac{i}{t} \times l$ at iteration i, and l is the original learning rate.

Appendix D. VGG-16 + CIFAR-10

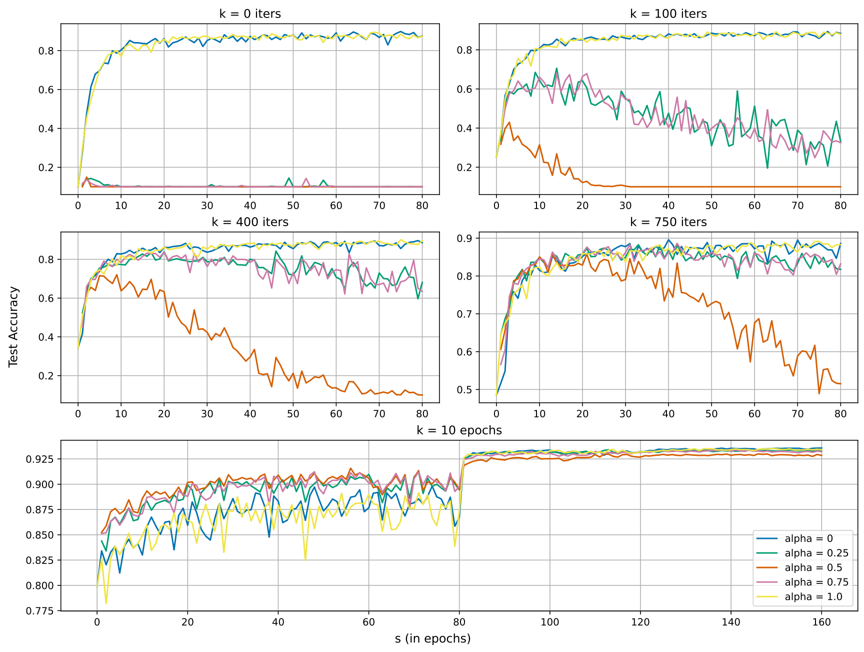

Figure 8: Progression of test performance of networks resulting from linear weight interpolation of child networks.

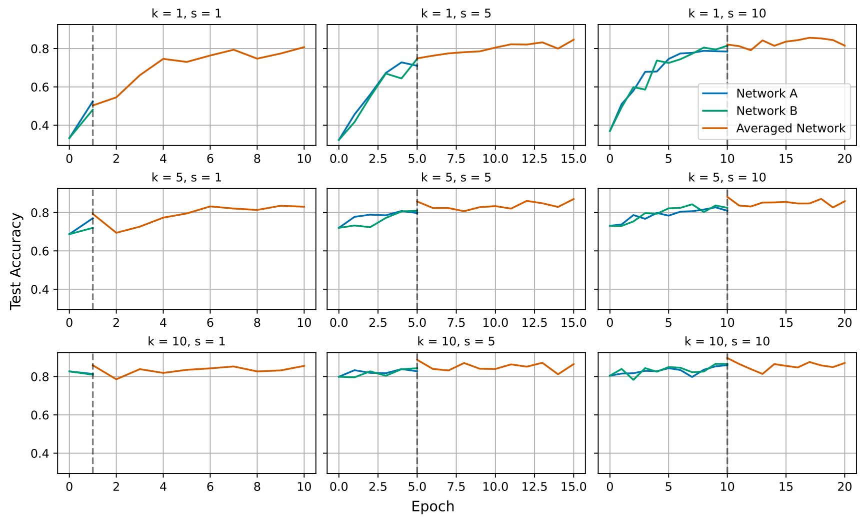

Figure 9: Test performance of child networks before interpolation and the weight-averaged network at, and after interpolation, for a variety of k and s values in epochs.

Appendix E. ResNet56 + CIFAR-100

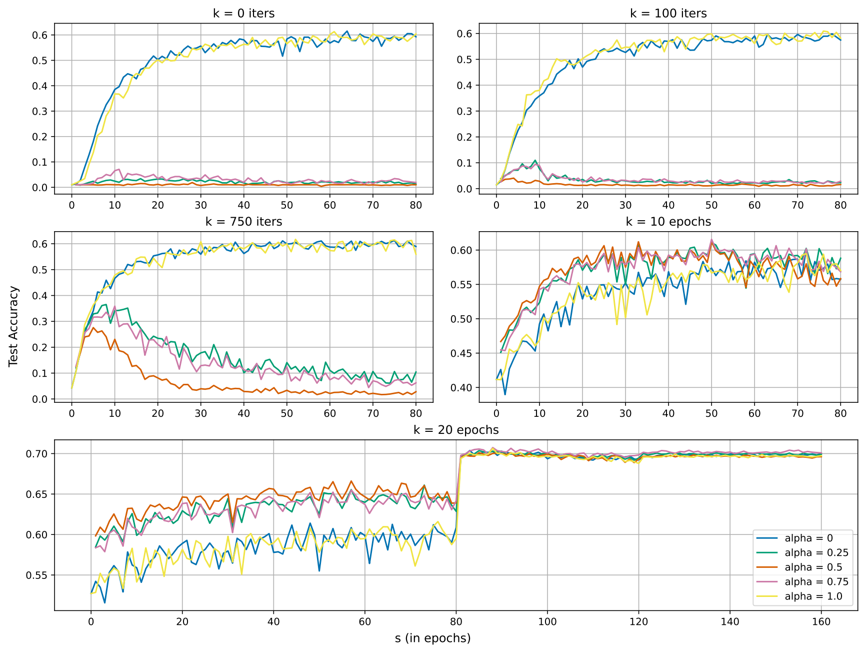

Figure 10: Progression of test performance of networks resulting from linear weight interpolation of child networks.

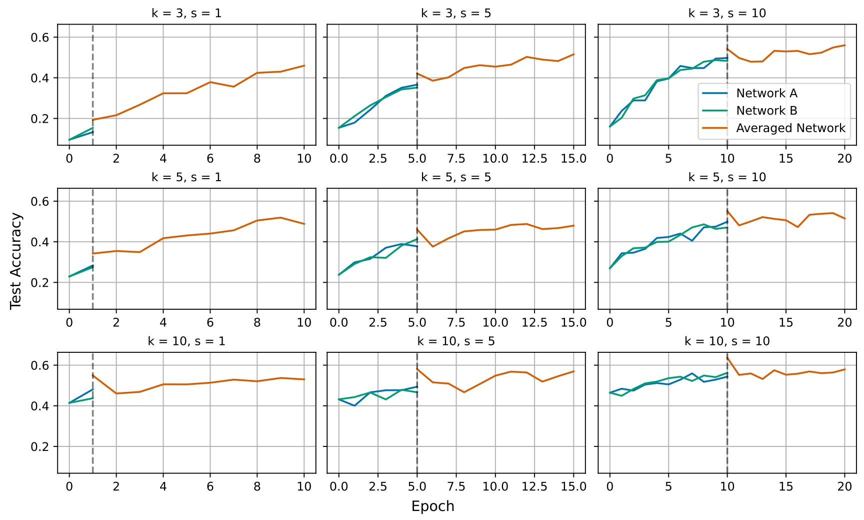

Figure 11: Test performance of child networks before interpolation and the weight-averaged network at, and after interpolation, for a variety of k and s values in epochs.

Appendix F. ResNet20 + CINIC-10

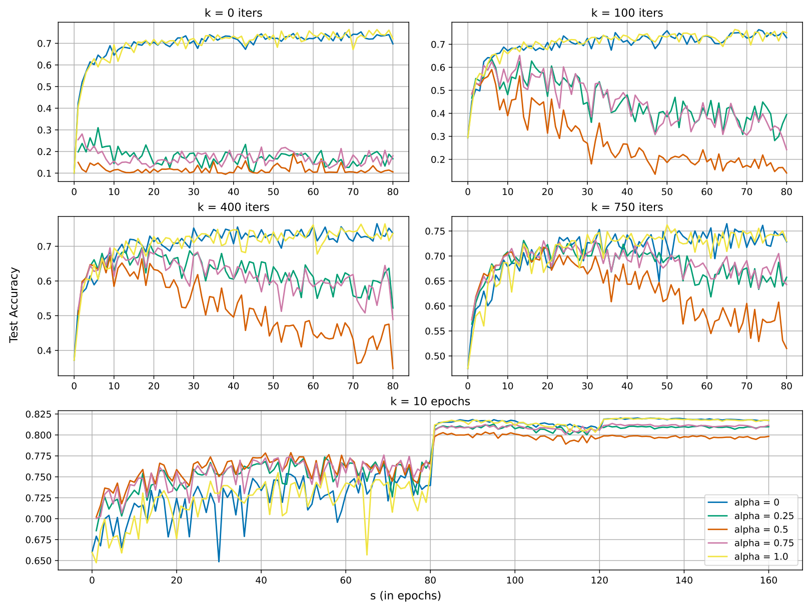

Figure 12: Progression of test performance of networks resulting from linear weight interpolation of child networks on CINIC-10 [3] with a ResNet20.

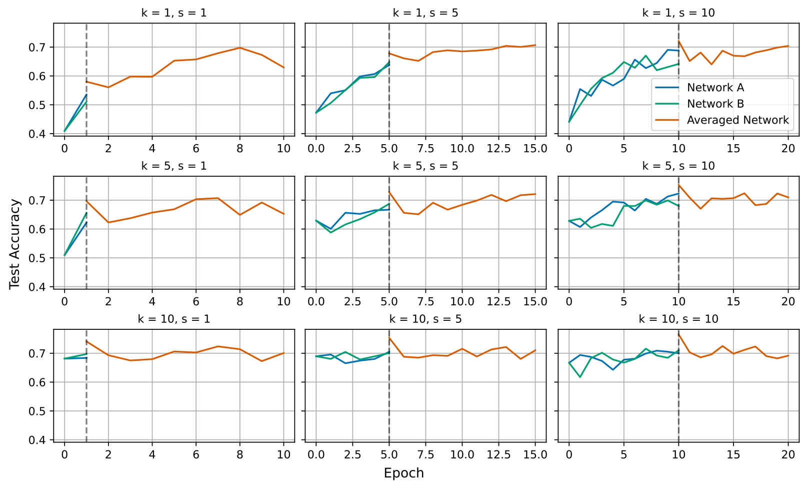

Figure 13: Test performance of child networks before interpolation and the weight-averaged network at, and after interpolation, for a variety of k and s values in epochs on CINIC-10 with a ResNet20.