Git Bi-sect: Dissecting Git Re-basin

Daniel Richards Ravi Arputharaj
KTH Royal Institute of Technology
drra@kth.se

Adhithyan Kalaivanan
KTH Royal Institute of Technology
adhkal@kth.se

Abstract

In this work, we reproduce the recent done by Ainsworth et al. [1] with a restricted class of models and datasets. We present phase transition like behaviour of the loss barrier on continued training, and better linearly connected modes than the original work. We also showcase the existence of single basin in MLP right from the start of training. We attribute these to the crucial role played by hyperparameters which remains to be extensively explored.

1 Introduction

Most deep neural networks exhibit permutational symmetry where the units of a hidden layer can be rearranged such that the resulting model is now at a different point in the weight space, without moving in the function space. Given the vast number of such degenerate solutions, Entezari et al. [2] propose a conjecture that most solutions found by SGD can be permuted to obtain linear mode connectivity. Ainsworth et al. [1] propose different methods to uncover such permutations that align two solutions and empirically demonstrate linear mode connectivity between them. Our work is an attempt at reproducing their central observations and results. We highlight the crucial role of optimization hyperparameters which influences the training trajectory and thus linear mode connectivity between solutions. Contrary to their conclusion that linear mode connectivity is an emergent phenomenon of training, we find permutations that align models with near zero loss barrier even before the end of the first training epoch. We also obtain near zero barrier for a standard width VGG-16 trained on CIFAR-10 and MNIST, thus overcoming one of their failure cases and showing that a wide model may not always be necessary. Our aim is to promote further investigation into the phenomenon of linear mode connectivity by sharing new observations which were not noted in the original work.

2 Code Repository

We implement the algorithms and experiments in PyTorch, and the code to reproduce our results can be found at https://github.com/the-nihilist-ninja/git-bisect

3 Related Work

We present a short overview of related works that explore mode connectivity, symmetries in neural networks and various approaches to model merging based on similarity of learnt representations.

Mode Connectivity: Garipov et al. [3] and Draxler et al. [4] concurrently found that optima in the loss landscape are connected by simple curves of low loss, which lead to various follow up works on mode connectivity. Shevchenko and Mondelli [5] present theoretical arguments for solutions found by SGD to be connected by low loss piece-wise linear paths. And the loss barrier along this path decreases with increasing model width. Tatro et al. [6] align model to be invariant to permutation by activation matching to get simpler low loss connecting curves. Then Benton et al. [7] extend beyond simple curves to simplexes that connect many independently trained models. Similar to

this work, Wortsman et al. [8] present methods to directly learn low loss simplexes instead of a single solution in one training run. Then we finally get to Ainsworth et al. [1] who use methods to remove the permutation symmetry of solutions and demonstrate linear mode connectivity on different architectures.

Permutation Symmetry: A variety of works explore the role of model permutation symmetry on the shape of loss landscapes. Brea et al. [9] show that permutation symmetry not only leads to a large number of global minima but also to first order saddle points along the path connecting them. Later, Simsek et al. [10] provide theoretical proof that all unconnected permutations of zero loss minima will be connected by a manifold by just adding one extra node to each hidden layer. Entezari et al. [2] put forth a conjecture that all SGD solutions can be permuted such that there will be a linear zero loss path connecting them. Based on this, Benzing et al. [11] compute permutation that can align two trained models by activation matching and use this permutation on the initial noise model to argue most that initialization lie in a single loss basin for MLP. Deng et al. [12] propose a novel methods that entirely circumvent this problem of arriving at degenerate solutions by training neural networks as principal eigenfunctions leading to representations that are ordered by their importance. Pittorino et al. [13] present a work that is very similar to what we replicate where the remove scale and permutation symmetry to demonstrate linear zero error path, instead of a zero loss path. They use a greedy version of weight matching to find the optimal permutation. Juneja et al. [14] find a different kind of linearly connected modes in transfer learning. They report clusters of linearly mode connected models finetuned from the same pretrained model, and that the clusters correspond to the generalization strategies adopted by these models. Works that go beyond permutation symmetry such as Meng et al. [15] who perform optimization on a scale invariant space is not extensively presented here. Lastly we draw attention to an earlier work by Fort and Jastrzebski [16] who model the high dimensional loss landscape as a union of low dimensional manifolds to show the existence of low loss subspaces connecting a set of optima. More importantly, they show that increasing regularization through weight decay, learning rate and model width lead to a wide *tunnel* or wide minima, i.e., higher distance to move from a solution so the loss significantly increases. Our observations on the influence of training hyperparameters may be linked to this, but needs further study to be confirmed.

Model Stitching and Merging: Most work that combine the weights of two independently trained models rely on the belief that they learn similar latent representations. Bansal et al. [17] demonstrate this by stitching different layers of even different architectures by introducing just a trainable intermediate layer. Concurrently Csiszárik et al. [18] stitch models of same architecture by introducing a single affine layer between them. Without explicitly investigating the representation similarities, Izmailov et al. [19] show that averaging weights of models can lead to better generalization. Godfrey et al. [20] provide a more general analysis of activation similarities by presenting the fundamental symmetry groups of a model. Lastly Moschella et al. [21] empirically demonstrate that different latent spaces are just a quasi-isometric transform of one another and leverage this to perform zero shot model stitching without introducing any intermediate layer.

4 Methods

The main conjecture put forth by Entezari et al. [2] is that most SGD solutions can be permuted to get an almost negligible loss barrier between them in the weight space. Ainsworth et al. [1] build on this conjecture and propose three methods to align two independently trained models, by permuting the hidden nodes without changing the function. We make a few changes to these algorithms with justification and explicitly note some details that the author deemed trivial.

Owing to computational constraints, we restrict our investigation to CIFAR-10 and MNIST datasets. And only train simple MLP and VGG-16 architectures, whose details are discussed in section 4.2

4.1 Algorithms

4.1.1 Activation Matching

While Ainsworth et al. find the optimal permutation using the hidden layer outputs after applying the activation function, we use outputs before the activation. We justify this choice by noting that ReLU activation function used in these models sets a large portion of the layer activations to zero and thus matching two trained models with this, results in the optimizer finding multiple optimal permutations

in each forward pass. Nevertheless, Benzing et al. [11] note that they found negligible difference between these two approaches.

4.1.2 Weight Matching

Ainsworth et al. do not specify the convergence criteria used in their permutation coordinate descent algorithm to stop the iterations. We choose the L_0 norm between optimal permutations from consecutive iterations for this purpose. In VGG-16, the transition from a convolutional layer to a fully connected layer through the flatten operation restricts the set of permutations possible for the fully connected layer. So we perform a convolution operation before obtaining the cost matrix as detailed in appendix A.1. This case isn't considered in the original work, as the VGG experiments are restricted to CIFAR. Further peculiarities of their model is presented in the following subsection.

4.1.3 Straight-Through Estimator

As our implementation is in PyTorch, we faced some set-back in the auto-grad functionality. PyTorch semantics works in an object-oriented manner, which meant modifying entities that are tracked by auto-grad cannot be done as easily as in JAX. Hence we use the experimental functorch [22] library for this purpose. We convert the Pytorch models to JAX-like functionals before the optimization step, which enables a simple implementation of the straight through estimator.

4.2 Architecture

4.2.1 MLP

We use a simple four-layer MLP whose hidden layers are 512 units wide for all experiments except the width ablation test. We also use the ReLU activation as in the original work.

4.2.2 VGG16

Ainsworth et al. substitute the batch normalisation layers in VGG-16 for layer normalisation. While this choice is not explicitly motivated in their work, we believe this may have been to avoid dealing with the batch statistics of the merged model, which is cannot be trivially merged like the weights. They also ignore the adaptive pooling layer at the end of convolutional blocks, because for CIFAR images this results in creating copies of the same values to match the output size of the pooling layer. This choice also meant they do not perform a flatten operation before the fully connected layer, and appropriately reduce the width of the linear layers. We too ignore the adaptive pooling operation, but our implementation of the computing and applying the optimal permutation is invariant to this. Details of how we handle this is explained in appendix A.1. We further perform all our VGG-16 experiments with the standard batch normalisation layers instead of layer normalisation as we observe no interesting difference between their results. To handle the batch statistics of the merged model, we reestimate them by performing one forward pass on the training data as shown by Izmailov et al. [19].

5 Data

Although the hypothesis being tested is applicable for neural networks in general, Ainsworth et al. [1] use popular computer vision datasets such as CIFAR-10, CIFAR-100, MNIST and ImageNet to demonstrate linear mode connectivity. Bound by computational constraints, we perform the core experiments on just CIFAR-10. Additionally, as VGG-16 trained on MNIST data is reported as a failure case where linear mode connectivity is not found, we investigate this particular case.

5.1 Pre-processing

CIFAR-10: To ensure that the results are robust to training hyperparameters, we augment the CIFAR10 training dataset using PyTorch implementation of AutoAugment based on Cubuk et al. [23], which is different from what is used by Ainsworth et al. [1]. We also normalize the input channels with a mean vector $(0.5, 0.5, 0.5)$ and standard deviation $(0.5, 0.5, 0.5)$.

MNIST: MNIST images are of size (28×28) which is lower than the minimum input size of VGG-16. So we resize the images to the required (32×32), and augment the data during training using PyTorch's RandAugument method based on Cubuk et al. [24]. The input is normalized with a mean of (0.1307) and standard deviation (0.3081).

6 Experiments and Findings

In our analysis, we restrict to VGG-16 and MLP trained on CIFAR-10 and MNIST using SGD with hyperparameters as reported in table 1. The test accuracy reached by these models on CIFAR-10 are $\sim$85% and $\sim$53% respectively. In MNIST, our VGG-16 model reaches an accuracy of $\sim$99%. VGG16 with batch normalisation is run twice to explore the role of hyperparameters on the loss barrier. We use the 1cycle learning rate policy introduced by Smith and Topin [25].

Table 1: Hyper parameters

(a) MLP on CIFAR-10

Parameter	Value
Learning rate	1e-3
Max learning rate	1e-1
# of epochs	40
Batch size	512
Momentum	0.9
Weight decay	5e-3

(b) VGG16 CIFAR-10 - Run 1

Parameter	Value
Learning rate	1e-3
Max learning rate	1e-2
# of epochs	100
Batch size	512
Momentum	0.9
Weight decay	1e-2

(c) VGG16 CIFAR-10 - Run 2

Parameter	Value
Learning rate	1e-5
Max learning rate	1e-1
# of epochs	100
Batch size	512
Momentum	0.9
Weight decay	1e-4

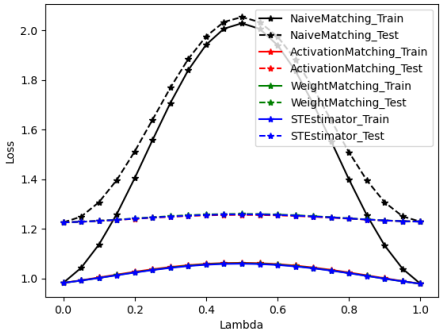
(d) VGG16 on MNIST

Parameter	Value
Learning rate	1e-3
Max learning rate	1e-2
# of epochs	100
Batch size	512
Momentum	0.9
Weight decay	1e-2

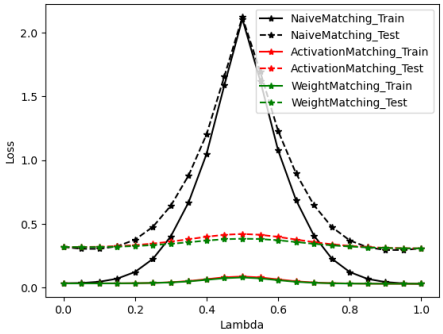
6.1 Core Experiments

The loss obtained by linearly interpolating between two models weights before and after alignment is shown in figure 1. From MLP's results we conclude that all three methods perform equally well. As the straight through estimator is computationally heavy, we skip it for the VGG model. From fig. 1b and fig. 1c, we note that training with a lower weight decay and learning rate results in a higher loss barrier and weight matching finds sub-optimal permutations. Observe that activation matching discovers a permutation whose barrier is close to the value reported in the original work. But with a different set of hyperparameters we manage to find near zero loss barriers for a standard width VGG network, which questions the conclusions on the effect of model width. This also prompted us to try one of the failure cases, VGG-16 with batch normalisation on MNIST as reported by Ainsworth et al. [1]. The result is shown in figure fig. 1d, once again achieving a near zero loss barrier, highlighting the impact of training hyperparameters. However, due to computational resource limitations we refrain from an exhaustive exploration on this hyperparameter dependence.

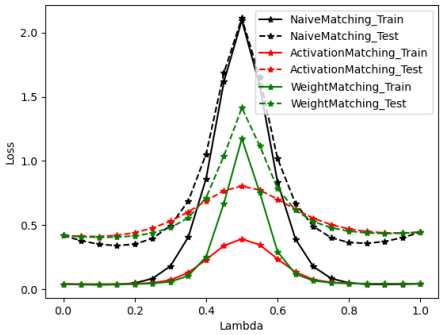
The difference between activation matching and weight matching under certain conditions led us to include activation matching while computing the epoch-wise loss barrier, whereas the original work only uses weight matching. While Ainsworth et al. conclude that linear mode connectivity is an emergent property of training, we observe that activation matching finds very low barrier solutions right from the start of training in the MLP case. Intriguingly this is not true while training VGG, even though the barrier found by activation matching is lower than weight matching. These results



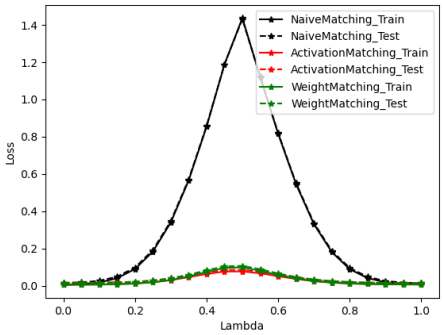
(a) MLP on CIFAR10



(b) VGG16-BN CIFAR10: Run 1



(c) VGG16-BN CIFAR10: Run 2



(d) VGG16-BN on MNIST

Figure 1: Loss plot at various intermediate points between two models

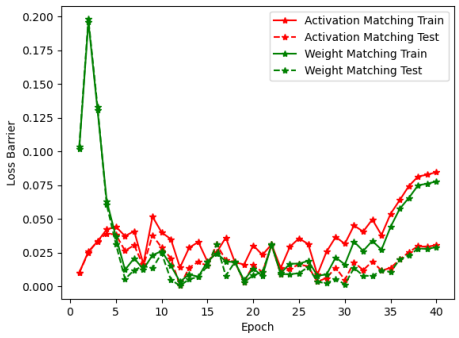
match those of Benzing et al. [11] who report no barrier on initialization using the permutation found at the end of training for MLP and their inability to observe the same in CNN. This raises interesting questions on the differences in the geometry of loss landscapes between different network architectures.

While Ainsworth et al. report the loss barrier obtained till epoch 25 for MLP, we observe a phase transition by training further. They attribute the variance in MLP to it being under-powered for CIFAR-10, but on continued training the loss barrier saturates, with the training loss barrier increasing significantly more than test. Once again the dynamics of VGG training is different, with a near constant zero barrier for most parts. Observe the small increase towards the end, which makes us unable to rule out the possibility that an MLP-like behaviour would emerge on training further. Such an equivalent experiment using the VGG network is not presented in the original work.

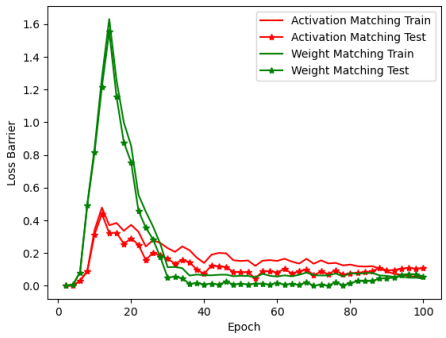
Lastly, we explore the effect of width using MLP, unlike Ainsworth et al. who show their results on VGG-16 and ResNet20. This choice is once again motivated by our limited computational resources. As noted earlier, we obtain a lower near zero barrier for the standard width VGG-16 contrary to their conclusion that thin models do not seem to obey linear mode connectivity. From fig. 2c, we confirm the decreasing loss barrier as model width increases. But note that value itself is very low for even thin models. So a strong conclusion cannot be made without exploring the role of training hyperparameters.

6.2 Additional experiments

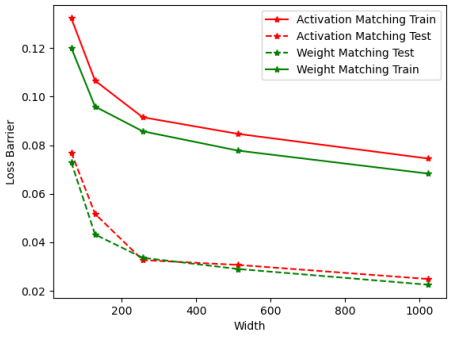
As activation matching finds zero barrier linear connectivity in MLP right from epoch 1, we look into the behaviour on random initialization and first 30 mini-batch updates. The result is presented in fig. 2d, from which we conclude that linearly connected zero loss barrier is indeed present from initialization for MLP.



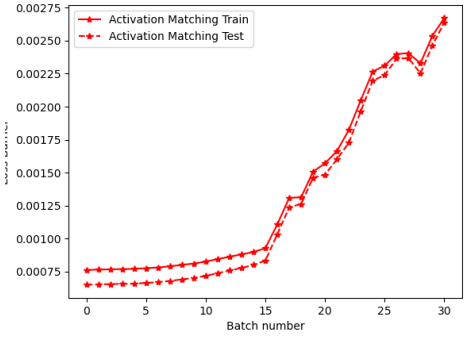
(a) MLP barrier epochwise



(b) VGG barrier epochwise



(c) MLP width ablation



(d) MLP barrier batch-wise

Figure 2: Ablation studies

Finally, we present our assessment on the efficacy of model merging as described in the original work. We independently train 5 MLP models on the CIFAR-10 dataset and merge by aligning them to one another before averaging their weights. We ask if this merged model effectively acts as the ensemble of the individual models. While the individual models reach accuracy of 0.5288, 0.5233, 0.5264, 0.5237, 0.5256 respectively, the merged model only reaches an accuracy of 0.5039. This is worse than individual models and also the ensemble which reaches 0.5293, i.e., better than any individual model. We also check how well calibrated the merged model is. From fig. 3, we see that merged model does not perform any better again. It is worth noting that all individual models show nearly the same performance both in accuracy and calibration, causing us to believe they might be lacking functional diversity. It would be interesting to observe how functional diverse models merge, by leveraging methods proposed by D'Angelo and Fortuin [26].

7 Challenges

As the influence of training hyperparameters like learning rate and weight decay were not discussed in the original work, we started out assuming the results were invariant to them as long as the model is well trained. But we realized this is false, when the reported loss barrier was not achieved even though the model performance was comparable. This lead to a lot of tweaking to understand the cause. And the author does not mention removing the adaptive pooling layer or resizing the fully connected layers of their VGG model. This was known only after contacting him via email. Lastly our choice of framework made the implementing the straight through estimator a lot harder than it had to be.

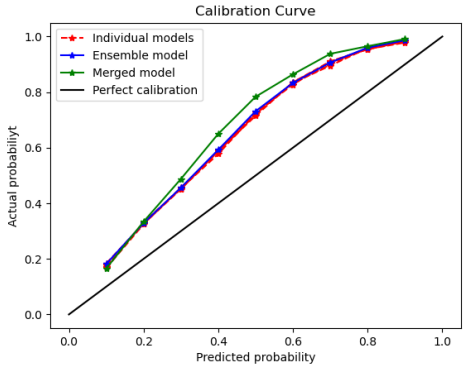


Figure 3: MLP merged model calibration plot

8 Conclusion

Through our experiments, we highlight that the phenomenon of linear mode connectivity is not confined to wide models or those with long training time, but may rather be influenced by a variety of interdependent factors that contribute to model regularization. We believe there is a lot of scope to more generally describe the conditions required for this. We also note the possible differences in loss landscapes of various network architectures. Lastly, connecting the seemingly different phases of the loss barrier through the epochs to the training dynamics and emergence of superposition as described by Elhage et al. [27] would also be an exciting area of research.

9 Ethical Consideration

Ainsworth et al. present interesting questions on the bias that might arise in a merged model as part of their ethics statement. Additionally, we note the need to research potential privacy concerns. While at first glance, model merging seems to protect the data seen by individual models, membership inference attacks [28] and training data extraction [29] is shown to be possible with just access to the model weights. If model merging requires the individual models to memorize their training data in order to perform well, this further increases their susceptibility. Lastly, we believe an investigation on whether desirable properties like interpretability are preserved through the merging process is also required.

10 Self Assessment

We believe that our project deserves an A grade and the arguments for that are presented in this section. We successfully reimplement all three methods and reproduce the main experiments using a restricted set of models and datasets. We also meet multiple bonus point requirements, including an implementation in PyTorch which differs from the author's JAX implementation. This choice required a clever work around for the straight through estimator, by converting the torch models to JAX-like functionals before each minimization step.

Our implementation also supports the standard VGG architecture with batchnorm layers and flatten operation. Ainsworth et al. [1] replace batchnorm with layernorm in their VGG model to avoid dealing with batch statistics while merging. Also by restricting their VGG-16 models to just the CIFAR datatset, and reducing the hidden units in the linear layer to (512) they remove the flatten operation. Including this would place further restrictions on the allowed permutations of the first linear layer, as we discuss in our work. Our negative results on the effect of width and training time, draw attention to the choice of training parameters and their impact on linear mode connectivity, which is overlooked. The demonstration of linear mode connectivity on MNIST with VGG is a strong case of the above. Lastly we provide informative additional works that are relevant to the phenomenon being studied.

References

[1] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2022. URL https://arxiv.org/abs/2209.04836.

[2] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=dNigytemkL.

[3] 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.

[4] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[5] Alexander Shevchenko and Marco Mondelli. Landscape connectivity and dropout stability of sgd solutions for over-parameterized neural networks. In *International Conference on Machine Learning*, pages 8773–8784. PMLR, 2020.

[6] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311, 2020.

[7] Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, pages 769–779. PMLR, 2021.

[8] Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *International Conference on Machine Learning*, pages 11217–11227. PMLR, 2021.

[9] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

[10] Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clément Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pages 9722–9732. PMLR, 2021.

[11] Frederik Benzing, Simon Schug, Robert Meier, Johannes von Oswald, Yassir Akram, Nicolas Zucchet, Laurence Aitchison, and Angelika Steger. Random initialisations performing above chance and how to find them, 2022. URL https://arxiv.org/abs/2209.07509.

[12] Zhijie Deng, Jiaxin Shi, Hao Zhang, Peng Cui, Cewu Lu, and Jun Zhu. Neural eigenfunctions are structured representation learners. *arXiv preprint arXiv:2210.12637*, 2022.

[13] Fabrizio Pittorino, Antonio Ferraro, Gabriele Perugini, Christoph Feinauer, Carlo Baldassi, and Riccardo Zecchina. Deep networks on toroids: Removing symmetries reveals the structure of flat regions in the landscape geometry. *arXiv preprint arXiv:2202.03038*, 2022.

[14] Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. *arXiv preprint arXiv:2205.12411*, 2022.

[15] Qi Meng, Shuxin Zheng, Huishuai Zhang, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. G-sgd: Optimizing relu neural networks in its positively scale-invariant space. *arXiv preprint arXiv:1802.03713*, 2018.

[16] Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in Neural Information Processing Systems*, 34:225–236, 2021.

[18] Adrián Csiszárik, Péter Kőrösi-Szabó, Ákos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34:5656–5668, 2021.

[19] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2018. URL `https://arxiv.org/abs/1803.05407`.

[20] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. *arXiv preprint arXiv:2205.14258*, 2022.

[21] Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. *arXiv preprint arXiv:2209.15430*, 2022.

[22] Functorch. URL `https://pytorch.org/functorch/stable/`.

[23] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[24] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[25] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[26] Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021.

[27] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[28] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[29] Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *arXiv preprint arXiv:2206.07758*, 2022.

A Appendix

A.1 Handling Convolutional to Fully Connected layer transition

Say we aim to align model A and model B which have filters of shape (C_l, C_{l-1}, H_f, W_f) at the last convolutional layer. The activation's would be flattened before applying the next fully connected layers whose weights are of shape $(C_{l+1}, C_l * H_a * W_a)$.

While computing the cost matrix for linear sum assignment in weight matching,

- Cost to compute permutation of the convolutional layer has the term $(W^A_{l+1})^T P_{l+1} W^B_{l+1}$ of shape $(C_l * H_a * W_a, C_l * H_a * W_a)$

  But due to restricted allowed permutations, the actual contributing cost matrix is obtained by a convolution of this matrix with an identity matrix of shape $(H_a * W_a)$ with a stride of $(H_a * W_a)$

- Cost to compute permutation of the first fully connected layer has the term $W^A_{l+1} P_l (W^B_{l+1})^T$.

  But weights are of shape $(C_{l+1}, C_l * H_a * W_a)$ and permutation is of shape $(C_l * C_l)$. The actual contributing cost should be obtained by first setting the permutations as $P_l \otimes \mathbf{I}_{(H_a * W_a)}$ and then multiplying the weights. Here $\otimes$ denotes the Kronecker product and $\mathbf{I}$ the identity matrix.