Random initialisations performing above chance and how to find them

Frederik Benzing[1] BENZINGF@INF.ETHZ.CH
Simon Schug[2] SSCHUG@ETHZ.CH
Robert Meier[1] ROMEIER@INF.ETHZ.CH
Johannes von Oswald[1] VOSWALDJ@ETHZ.CH
Yassir Akram[1] YASSIR.AKRAM@INF.ETHZ.CH
Nicolas Zucchet[1] NZUCCHET@INF.ETHZ.CH
Laurence Aitchison[3, *] LAURENCE.AITCHISON@BRISTOL.AC.UK
Angelika Steger[1, *] STEGER@INF.ETHZ.CH

[1] *Department of Computer Science, ETH Zurich, Switzerland*
[2] *Institute of Neuroinformatics, ETH Zurich, Switzerland*
[3] *Department of Computer Science, University of Bristol, UK*
[*] *Equal contribution*

Abstract

Neural networks trained with stochastic gradient descent (SGD) starting from different random initialisations typically find functionally very similar solutions, raising the question of whether there are meaningful differences between different SGD solutions. Entezari et al. recently conjectured that despite different initialisations, the solutions found by SGD lie in the same loss valley after taking into account the permutation invariance of neural networks. Concretely, they hypothesise that any two solutions found by SGD can be permuted such that the linear interpolation between their parameters forms a path without significant increases in loss. Here, we use a simple but powerful algorithm to find such permutations that allows us to obtain direct empirical evidence that the hypothesis is true in fully connected networks. Strikingly, we find that two networks already live in the same loss valley at the time of initialisation and averaging their random, but suitably permuted initialisation performs significantly above chance. In contrast, for convolutional architectures, our evidence suggests that the hypothesis does not hold. Especially in a large learning rate regime, SGD seems to discover diverse modes.[1].

1. Introduction

Neural networks trained from different initialisations typically have similar decision boundaries [34], their predictions agree on a large fraction of data [25] and even their internal representations are similar [17, 22]. This raises the question if all trained instances of neural networks are created equal. Entezari et al. [8] recently proposed one way to formalise this idea and hypothesised that any two solutions found by stochastic gradient descent (SGD) live in the same loss valley when accounting for the permutation invariance of neural networks. More precisely, they suggest the following hypothesis.

Hypothesis 1 (All SGD solutions are created equal, Entezari et al. [8]) *For two solutions θ_1, θ_2 obtained by SGD, there is a permuted instance of the first solution $\pi(\theta_1)$ so that there is no (significant) increase in loss along the linear interpolation between $\pi(\theta_1)$ and θ_2.*

1. Code available at: https://github.com/freedbee/permuted_initialisations

© F. Benzing[1], S. Schug[2], R. Meier[1], J. von Oswald[1], Y. Akram[1], N. Zucchet[1], L. Aitchison[3, *] & A. Steger[1, *].

If Hypothesis 1 is true, this would greatly improve our understanding of the loss landscape of neural networks, likely impact ensemble learning [12] as well as network sparsification [10] and it would potentially allow to improve optimisation and generalisation [15] in neural networks. Here, we investigate this hypothesis by explicitly constructing permutations with the desired properties, leading to a range of novel findings that elucidate the structure of neural loss landscapes.

- We find strong, direct evidence in support of the hypothesis for fully connected networks of moderate and large width on standard datasets.
- For fully connected networks, we find that an even stronger version of the hypothesis is true: The networks already live in the same loss valley at initialisation. In particular, averaging the random initialisations $\pi(\theta_1^{\text{init}}), \theta_2^{\text{init}}$ using the permutation revealed through training, gives a network performing significantly above chance.
- For convolutional neural networks (CNNs), our evidence suggests that there is a notable loss barrier between solutions also after accounting for permutation invariance. In particular, and interestingly, in the large learning rate regime, SGD seems to explore a genuinely diverse set of solutions.

We discuss exciting concurrent work [1] as well as generally related work on permutation invariance and mode connectivity in Appendix A .

2. Background

We would like to evaluate whether two sets of parameters θ_1, θ_2 lie in the same loss valley after accounting for permutation invariance. To this end, we seek a permutation π^* of the parameters θ_1 that minimizes the loss barrier between the two sets of parameters after applying the permutation, i.e. we would like to solve

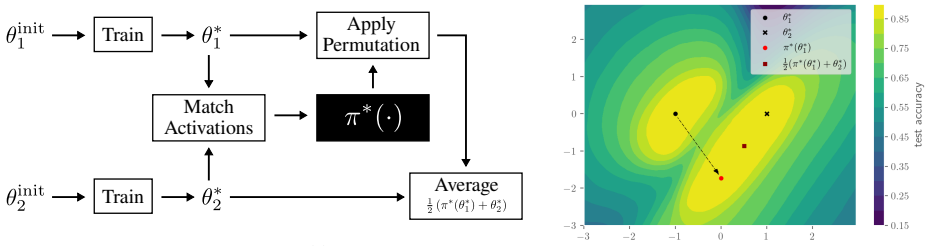
$$\min_{\pi} \text{Barrier}(\pi(\theta_1), \theta_2) \tag{1}$$

$$\text{Barrier}(\theta_1, \theta_2) := \max_{\lambda \in [0,1]} \ell\big(\lambda\theta_1 + (1-\lambda\theta_2)\big) - \Big(\lambda\ell(\theta_1) + (1-\lambda)\ell(\theta_2)\Big) \tag{2}$$

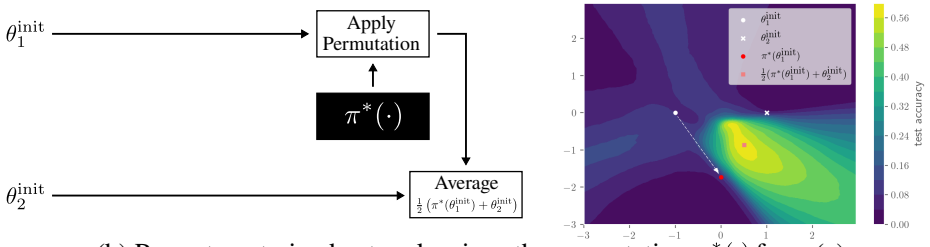
While this problem is computationally intractable, a simple heuristic allows us to find good permutations in practice. We relax the problem by considering the neuron activations of each layer separately. For each layer, we then find a permutation that makes the resulting neuron activations as similar as possible.

Formally, given a set of d data points, the neuron activations of the networks parameterized by θ_1 and θ_2 are given by the matrices $A, B \in \mathbb{R}^{d \times n}$, where n denotes the number of neurons in the hidden layer (or the number of channels for CNNs). We define a similarity function $k(\cdot, \cdot)$ which measures for any pair of neurons i, j how similar their activation vectors A_i, B_j are and specify our surrogate objective as finding the permutation π which maximizes the summed similarity between all matched neuron activations $s(\pi)$

$$s(\pi) := \sum_{i=1}^{n} k(A_{\pi(i)}, B_i) \tag{3}$$



(a) Find permutation $\pi^*(\cdot)$ by matching activations from two trained networks



(b) Permute untrained networks given the permutation $\pi^*(\cdot)$ from (a)

Figure 1: **Training procedure and accuracy heat map of plane spanned by three networks, $\theta_1, \theta_2, \pi(\theta_1)$, for trained networks (top) and at initialisation (bottom).** After training two independently initialised networks, performance decreases when the trained weights are interpolated. This barrier vanishes after applying the permutation found by matching activations. Strikingly, when applying the permutation to the untrained initialisations, interpolating between the two results yields significantly above chance performance. Data from a fully connected network trained on Fashion MNIST.

Maximizing $s(\pi)$ corresponds to finding a maximum matching in a bipartite graph and can be solved efficiently using the Hungarian algorithm. We explored different variants of this idea which all performed similarly. Unless noted otherwise we use the cosine similarity as similarity function k and consider the neuron activations after the non-linearity. Finding permutations based on this idea has already been explored in [5, 22] and re-described e.g. in [1, 33, 35, 38].

3. Results

3.1. Fully connected networks

For the first set of experiments, we will focus on fully-connected networks with varying width and depth trained on MNIST [21], Fashion MNIST [37] or CIFAR 10 [19]. In the main paper we will show results on Fashion MNIST; results on other datasets are qualitatively similar and can be found in Appendix G.

The training and evaluation procedure is simple and outlined in Figure 1(a). We train two networks independently, for 5 epochs each with AdamW [16, 23] with PyTorch [29] default settings. We then compute the permutation making the networks' activations as similar as possible. Subsequently, we evaluate networks along the linear interpolation between the two networks, either using their unpermuted or permuted versions.

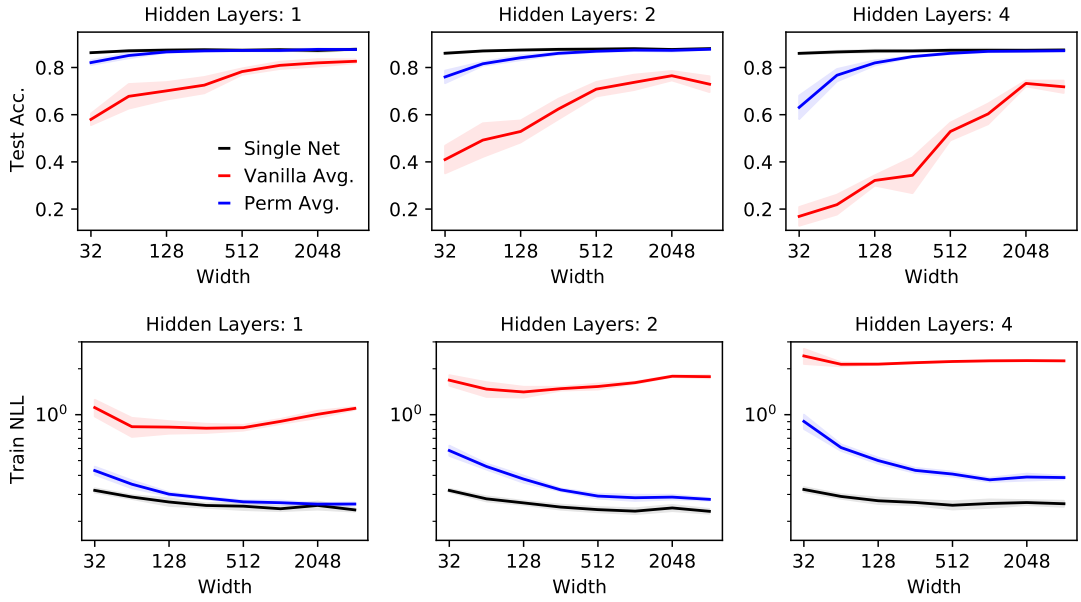


Figure 2: **Loss barrier reduces after permutation and depends on network depth and width.** All networks are fully connected and trained on Fashion MNIST for five epochs with AdamW. First, we plot the performance of such a network ("Single Net", black line). Then, we plot the performance when the parameters of two trained networks are averaged, with ("Perm Avg.", blue) or without ("Vanilla Avg.", red) being permuted. Thus, the loss barrier is the difference between the "Single Net" line (black) and the "Perm Avg" line (blue). Shaded region show twice the sem (approx. 95% confidence interval) over 6 seeds.

Trained networks lie in the same loss valley after permutation. Figure 2 shows the results of these experiments. The loss barrier between solutions decreases dramatically after accounting for permutation invariance. This decrease confirms that permutation invariance is important for understanding mode connectivity. Furthermore, by sufficiently increasing the width of the networks the loss barrier is eliminated completely by accounting for permutation invariance. Thus, in wide networks, this directly confirms the hypothesis of Entezari et al. [8], that there is no significant loss barrier between SGD solutions after taking permutation invariance into account.
However, in narrower networks, we did not find permutations that eliminate the loss barrier suggesting that the two solutions are genuinely different even when accounting for permutation invariance. That said, we cannot exclude the possibility that our permutations are suboptimal, and that "better" permutations would eliminate the loss barrier. Indeed, a thorough falsification of the hypothesis in narrower networks would require an exhaustive (and therefore infeasible) search over the space of permutations.

Averaged, random initialisations perform above chance after permutations. We have seen that trained networks usually live in the same, or at least similar, loss valleys when accounting for

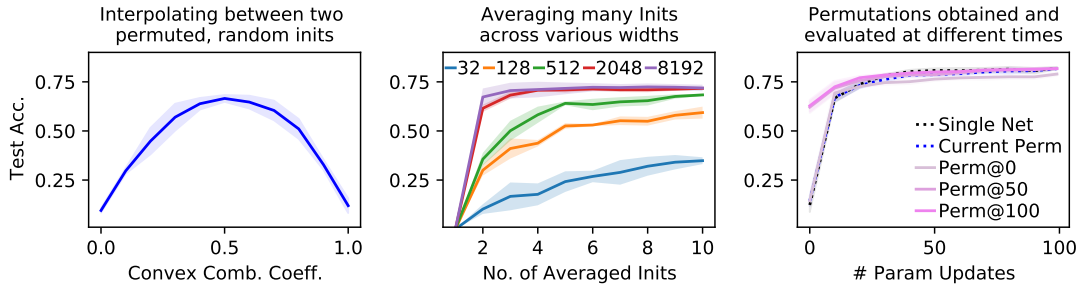


Figure 3: **Permutations found after training project random initialisations to same loss valley. Left:** Network with two hidden layers of 2048 neurons each, trained on Fashion MNIST. Left- and rightmost point correspond to two random initialisations. The average of the two permutations performs significantly above chance, demonstrating that, after the permutation, the initialisations lie in the same loss valley. **Mid:** Networks with one hidden layer and different widths. For $n > 2$ networks, all networks are permuted to be similar to the first network, and then averaged with equal weights. Averaging increasing numbers of random initialisations gradually improves performance. **Right:** Same architecture as left panel, but permutations are computed and applied at different points at training. We match, permute and average the activations after each parameter update ("Current Perm", dashed blue). Additionally we apply the permutations found after 0, 50 respectively 100 parameter updates at different times in training ("Perm@x", shades of pink). Permutations that reveal that initialisations live in the same loss valley can also be found without training the networks, see Appendix B.

permutation invariance. Next, we ask how much training is required until the networks enter a shared loss valley.

To investigate this, we conduct experiments as outlined in Figure 1(b), i.e. we take two random initialisations, train them with independently shuffled data, determine a permutation that makes the trained networks similar and then apply this permutation to the (untrained) initialisation. Figure 3(left) shows the interpolation between those permuted initialisations and shows a rather surprising discovery: Already the initialisations live in a shared loss valley (up to a permutation), and simply averaging the permuted initialisations gives significantly above chance performance. Figure 3(mid) shows that performance of these random initialisations can be further improved by averaging more than two of them. Figure 3(right) also shows that comparatively little training is required to reveal the shared loss valley between random initialisations.
We also note that it is possible to use the training data to find permutations that put initialisations into the same loss valley without actually training the networks, see Appendix B.

Summary. Overall, these findings give a significantly improved understanding of the loss landscape of fully-connected networks. They reveal that, up to permutation, fully-connected feedforward networks live in the same loss valley at initialisation. Moreover, even as the networks start specialising during training, the loss barrier separating any two SGD solutions remains small.

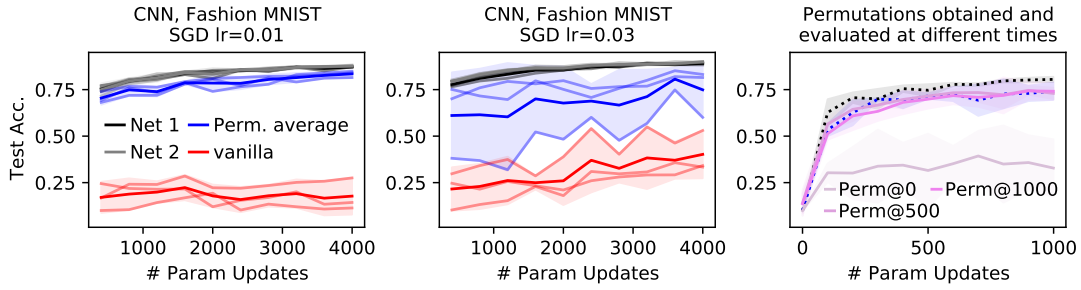


Figure 4: **Diversity of SGD solutions depends on training regime for CNNs & random initialisations reside in different loss valleys. Left & Mid:** Two identical CNNs but different initialisations (Net 1 and Net 2) trained with different learning rates. With a smaller learning rate (left), the loss barrier between different solutions is very small after accounting for permutation invariance; whereas with a larger learning rate (mid) the loss barrier increases notably. This suggests that SGD explores more diverse solutions with larger learning rates. Analogous behaviour can be observed in other settings, e.g. for CIFAR 10 and AdamW, see Appendix G. **Right:** Different permutations applied across different times, analogous to Figure 3 (right). This plot also shows that for CNNs the average of permuted initialisations does not perform notably above chance.

3.2. Convolutional networks

Next, we investigate simple CNNs purely composed of convolutional layers with no skip connections and a linear readout layer (see Appendix F for network and training details).

We again find that accounting for permutation invariance significantly reduces the loss barrier between SGD solutions, in a wide range of settings and as exemplified in Figure 4; however a notable barrier remained in most of our settings, including wide networks. This is also consistent with results from more complex architectures, see Fig. 3 in [35].

In addition, we found that the loss barrier in CNNs is significantly more sensitive to training details, like the learning rate.[2] One example of this behaviour is shown in Figure 4 (left&mid), where with a larger learning rate the barrier between solutions increases. We emphasise that this behaviour can be observed across different convolutional architectures, datasets and also optimizers (AdamW as well as SGD+momentum). In all these settings, the loss barrier increases for large learning rates, which are at the edge of training stability (occasionally, runs with these high learning rates fail to converge). These results suggest that for CNNs trained with large learning rates, SGD explores diverse loss valleys, which differ also after accounting for permutation invariance.[3] Moreover, we note that for CNNs, different initialisations do not appear to come from a shared loss valley, see Figure 4 (right).

Summary. These results reveal two interesting differences between fully-connected and convolutional architectures. Firstly, in fully-connected networks already the initialisations live in a shared

2. As a side effect, it is harder to compare the loss barrier between architectures of different width and depth, since for each architecture the choice of training setup has a non-negligible effect.
3. We did not find similar behaviour for fully connected networks, despite looking for it.

loss valley, while this does not seem to be the case for CNNs. Secondly, CNNs exhibit a richer training behaviour and in particular for large learning rates, close to training instability, they appear to explore genuinely different loss valleys, while solutions found by SGD in fully connected network are more similar.

Acknowledgements

Simon Schug was supported by grant no. PZ00P3_186027 from the Swiss National Science Foundation. Robert Meier was supported by grant no. CRSII5 173721 of the Swiss National Science Foundation. Johannes von Oswald was funded by the Swiss Data Science Center (J.v.O. P18-03). Nicolas Zucchet and Yassir Akram were supported by ETH Research Grant no. ETH-23 21-1.

Appendix A. Related work

A.1. Concurrent related work

Concurrently, Ainsworth et al. [1] explored a similar approach to investigate Hypothesis 1. In particular, the base algorithm to find permutations is analogous to ours and that in prior work [5, 22]. While both papers agree in their key findings, they also highlight different features. The main differences are briefly described below:
Ainsworth et al. [1] present results in a ResNet18 on CIFAR 10, showing that the loss barrier also decreases (and starts to disappear) with increasing width. They also show a simple counterexample to the hypothesis for a small network with artificial data and explore merging networks trained on different subsets of the data. In contrast, our contributions contains a systematic investigation into width- & depth-dependence of the loss barrier in fully-connected networks and we highlight the learning rate dependence of the loss barrier in CNNs. Moreover, we showcase that in fully connected networks, already the untrained initialisations live in the same loss valley.

A.2. Prior related work

The question how similar different neural networks are has inspired research from a variety of angles. For example, Li et al. [22] train independent copies of CNNs and investigate how well individual neurons and groups of neurons from one network correspond to groups of different neurons; part of their methodology also relies on computing matchings analogously to our approach. Kornblith et al. [17] investigates similarity of layer representations drawing on work from neuroscience (e.g. [18]) and machine learning [20]. Their similarity measure is theoretically well-motivated and suggests that different networks learn considerably more similar representations than suggested by similarity measures of prior work [27, 30, 36]. Mehrer et al. [26] also study network similarity with methods from neuroscience and additionally conduct insightful experiments on how inter- or intra-class variance contribute to differences between networks. All [17, 22, 26] find that representations agree very strongly in early layers and decay to some extent for deeper layers. This is consistent with our findings in Figure 2 suggesting that deeper networks are separated by higher loss barriers.

In addition, our work is closely connected to mode connectivity. It was first shown theoretically that any two modes are connected by a (potentially complex, non-linear) path of low loss in networks with one hidden layer [11]. This was confirmed in a range of empirical settings [7, 12]. Tatro et al. [35] show that these paths can be simplified or their loss barrier lowered when taking permutation invariance into account. They find permutations similarly as us. Thus, their work is overall similar in spirit to ours, with the difference that we investigate linear paths between modes and how architecture and training choices affect mode connectivity. Theoretically, [4, 32] investigate the connectivity between permutations of a given mode. We note that this is different from our objective to determine whether independently obtained modes are (approximately) permutations of one another. Additionally, there is work developing network-/kernel-representations which are invariant to permutations and rotations [2]. Linear mode connectivity has also been investigated in different contexts, e.g. focusing on linear connections between different points from a training trajectory of neural networks [13]; focussing on linear connections between networks trained on independent data samples from the same initialisation [28]; relating linear mode connectivity to flatness and generalisation [15], and relating it to sparsity and the lottery ticket hypothesis [9, 10].

Appendix B. Finding good permutations at initialisation without training

In the main paper we described how a permutation obtained by training networks and matching their features leads to averaged initialisations that perform above chance. Here, we briefly note that such permutations can also be obtained without training the networks and describe how this can be done.

As a first observation note that for our desired permutation π, the loss decreases when we start at θ_2 and move towards $\pi(\theta_1)$, i.e. in the direction of $v_\pi = \pi(\theta_1) - \theta_2$. This suggests that the direction v_π should be (negatively) aligned with the gradient at θ_2. Thus, we can try to find permutations that minimize the dot product $v_\pi \cdot \frac{\partial L(\theta)}{\partial \theta}|_{\theta=\theta_2}$. This can be formalised as a minimum matching problem again and thus be solved with the Hungarian algorithm.
The idea described above does not yet give good permutations and we need two additional ideas.

- First, and most importantly, we replace the vanilla gradient $\frac{\partial L(\theta)}{\partial \theta}|_{\theta=\theta_2}$ by the update direction of a more powerful optimizer. We used FOOF [3], which was inspired by obtaining a better understanding of KFAC [6, 24].
- Second, we symmetrise the loss. I.e. (using the vanilla gradient rather than more advanced optimizers for notational simplicity) we minimize

$$\min_{\pi} \; (\pi(\theta_1) - \theta_2) \cdot \frac{\partial L(\theta)}{\partial \theta}|_{\theta=\theta_2} + (\pi^{-1}(\theta_2) - \theta_1) \cdot \frac{\partial L(\theta)}{\partial \theta}|_{\theta=\theta_1} \tag{4}$$

We note that this gives permutations with very similar performance (at initialisation) as the permutations described in the main text. Moreover, in the case of CNNs this algorithm also does not find permutations which put different initialisations in the same loss valley, further supporting our finding from the main paper that CNN initialisations live in distinct loss valleys.

Appendix C. Finding permutations for second-order Taylor expansions around the loss

The algorithm described in the preceding section (almost) amounts to finding a permutation that minimizes a first-order Taylor expansion of the loss around θ_2 evaluated at $\frac{1}{2}\pi(\theta_1) + \frac{1}{2}\theta_2$.[4] Thus or otherwise motivated, one might be tempted to find permutations that minimize higher-order Taylor expansions of the loss around θ_2 evaluated at $\frac{1}{2}\theta_2 + \frac{1}{2}\pi(\theta_1)$ both for initialisations or trained networks θ_2.
If one uses KFAC [24] to obtain a computationally tractable approximation of the second-order expansion of the loss around θ_2, and then computes the permutations sequentially layer-by-layer, the minimization problem to find good permutations turns out to be equivalent to the quadratic assignment problem (and thus NP hard). We experimented with convex relaxations of this problem over the Birkhoff polytope but failed to find better permutations than with the simpler method described in the main paper.

Appendix D. Variants of feature matching and interpolation

We experimented with different similarity functions $k(\cdot, \cdot)$ for measuring the similarity between neuron activations, in particular ℓ_2-distance and cosine similarity. We also investigated extracting

4. We choose the midpoint between $\pi(\theta_1)$ and θ_2 here since empirically, this is a very good approximation of the point along the linear interpolation that maximizes the loss barrier.

neuron activations both before and after the non-linearity. Broadly speaking, all variants performed very similarly. In addition, we experimented with combining the permutations with learned / non-learned "small" rotations. This often decreased the loss barrier slightly, but did not affect results qualitatively. Moreover, we tried interpolating between each layers' weights using a rotation, rather than a linear path, but found that this significantly increased the loss barrier.

Finally, we tried matching weights instead of neuron activations. Concretely, rather than finding a permutation that maximizes similarity between activation vectors, we found a permutation that maximizes similarity between weight vectors. In principle, for every layer except the first and the last, this requires computing two permutations. To circumvent this problem, we simply started with the first layer, and moved upwards step by step. Matching weights performed slightly worse than matching activations, i.e. the loss barrier increased.

Appendix E. Recap of permutation invariance

It is well known that neural networks are permutation invariant. To make this term precise, consider a fully connected network consisting of a sequence of weight matrices $W_1, \dots, W_\ell$, where W_i has dimensions $n_{i-1} \times n_i$, and a pointwise non-linearity σ (for simplicity of notation, we assume that each layer is followed by the same non-linearity, but this assumption can easily be dropped). For a given input X, the output of the i-th layer A_i is defined iteratively as

$$A_0 = X \tag{5}$$

$$A_i = \sigma(W_i A_{i-1}) \quad \text{for } 1 \leq i \leq \ell \tag{6}$$

and the output of the neural network is simply the output of the last layer A_ℓ.

If we are given a sequence of permutation matrices $P_0, \dots, P_\ell$, where P_i has dimensions $n_i \times n_i$ and where the first and the last permutation matrix P_0, P_ℓ are the identity, then the permuted neural network consisting of the weight matrices

$$\tilde{W}_i = P_i^T W_i P_{i-1}$$

computes precisely the same output as the network consisting of weight matrices W_i (since non-linearity and permutation commute and since $P_i^T P_i = \mathrm{Id}$). Thus the input-output function specified by the network is invariant to permutations as described above.

While we used fully connected networks above, permutation invariance straightforwardly applies for example to convolutional networks, where channels can be permuted.

Appendix F. Experimental details

The term "SGD" in the paper usually refers to SGD with the usual 0.9 heavy-ball momentum, but we obtained qualitatively similar results for SGD without momentum. We used a batch size of 100 in all experiments, but we found similar results across a range of batch sizes. The permutations are usually computed using the activations of 10000 images, and we verified that the quality of permutations saturates already for smaller numbers of images. We used Kaiming-He init and ReLU nonlinearities for intermediate layers. The final layer is followed by a softmax, in combination with the cross-entropy loss.

F.1. Fully connected networks

Networks presented here are trained for 5 epochs with AdamW PyTorch [29] default settings, except Figure 3 mid-panel, where permutations were obtained by training for one epoch. We note that for narrow networks, permutations that obtain even better performance when used to average initialisations can be obtained by training shorter.

F.2. CNNs

All CNN architectures use a sequence of convolutional filters and 2x2 average pooling layers, based on standard CNN design practices. These layers are finally followed by a global average pooling layer and one fully connected layer.

Based on this general design, we trained two kinds of CNN architectures, one of type

$$k - k - \text{avg} - 2k - 2k$$

where "avg" denotes a 2x2 average pooling layer and the remaining numbers denote the number of (hidden) channels in the convolutional layers; and another one of type:

$$k - \text{avg} - 2k - \text{avg} - 4k.$$

We used $k = 32, 64, 128, 256$

The CNN in Figure 4 is of type $128 - 128 - \text{avg} - 256 - 256$, but we observed similar behaviour across different optimizers, architectures and datasets. One more example for CIFAR10, AdamW and type $64 - 64 - \text{avg} - 128 - 128$ is shown in Figure 10.

Appendix G. Additional figures

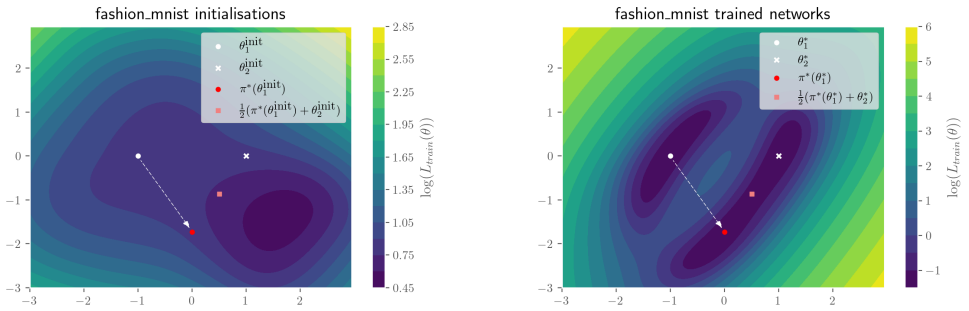


Figure 5: Corresponding Train Loss Heat Map to the right part of Figure 1.

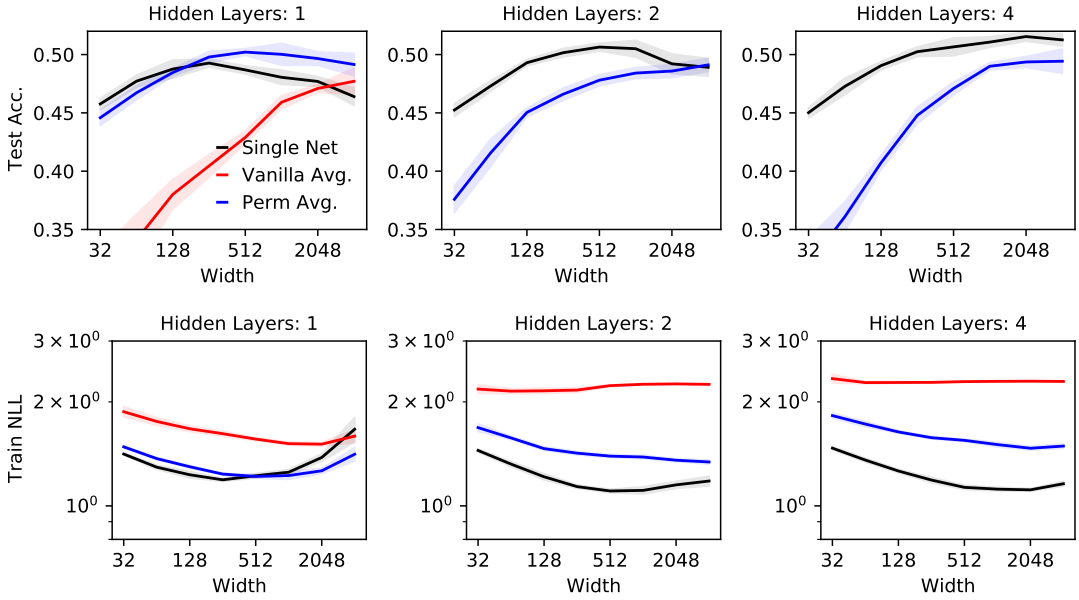


Figure 6: Analogous to Figure 2, but on CIFAR 10.
Loss increases at very high width, because in this case lower learning rates would be optimal, but we stick to AdamW default settings.

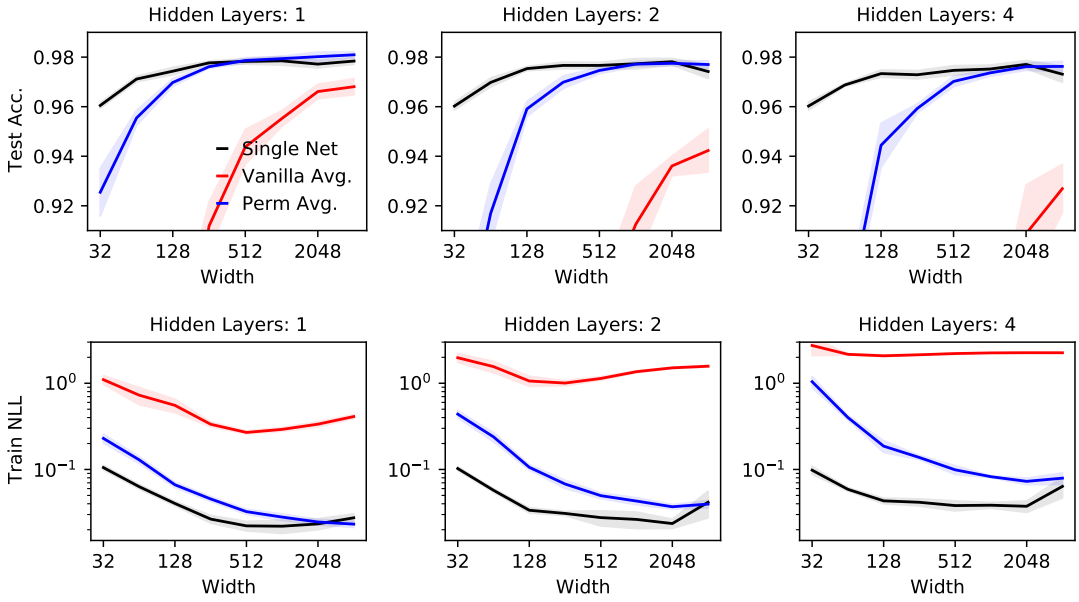


Figure 7: Analogous to Figure 2, but on MNIST.
Loss increases at very high width, because in this case lower learning rates would be optimal, but we stick to AdamW default settings.

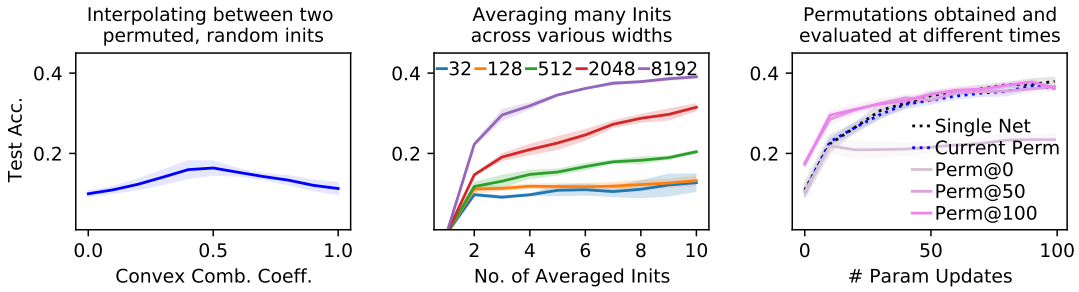


Figure 8: Analogous to Figure 3, but on CIFAR 10.

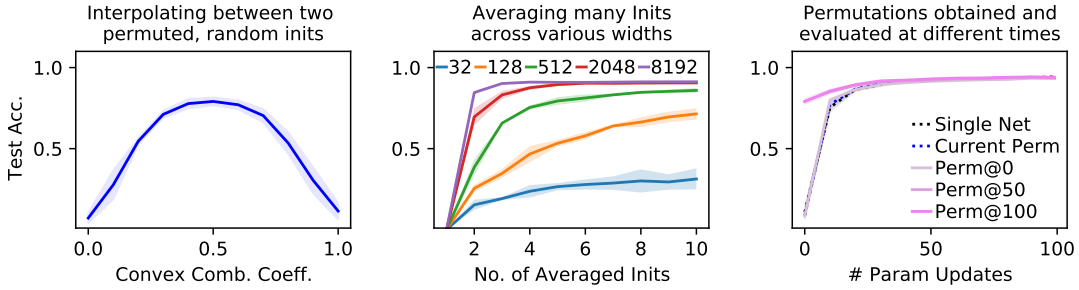


Figure 9: Analogous to Figure 3, but on MNIST.

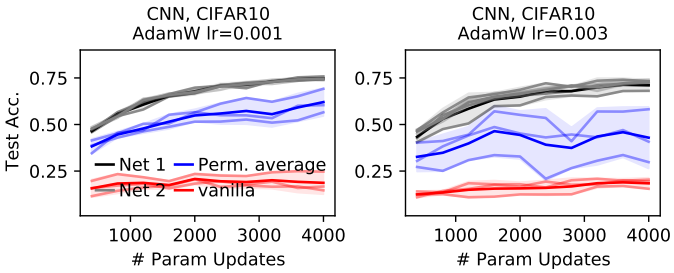


Figure 10: Analogous to Figure 4, but on CIFAR10 with AdamW.

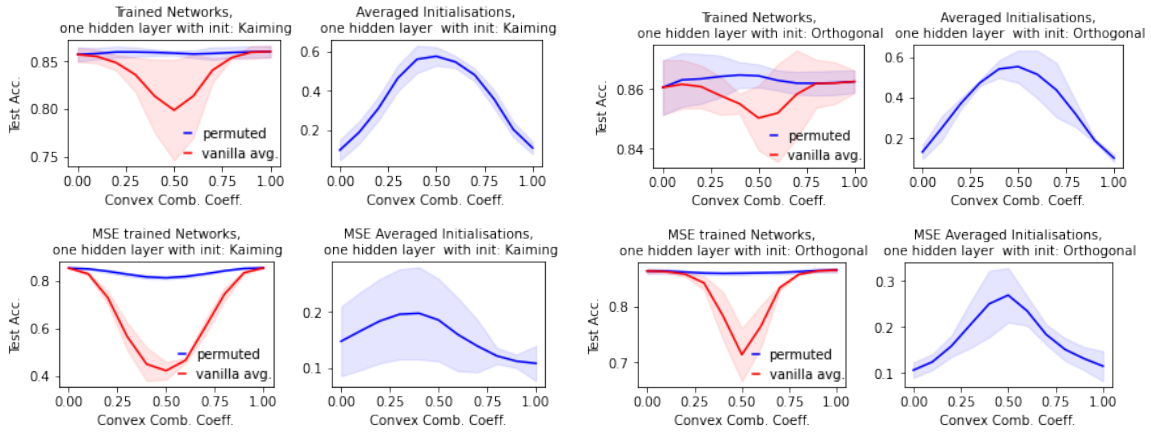


Figure 11: Additional results on Fashion MNIST when varying the initialization scheme and loss function. *Top row*: The effect of vanilla and permuted averaging, at initialization and after training, for two different initialization schemes. We test Kaiming-He (as in the main text) [14], Xavier (not shown) and Orthogonal [31] initialization and find that the mode connectivity remains linear for the different initializations. Behaviour with Xavier initialization is very similar to the Orthogonal one and therefore omitted. *Bottom row*: When turning to the mean squared error (MSE) loss instead of the cross-entropy loss, we observe worse performance for vanilla averaging before and after training, for the two initializations, while the networks stay linearly connected after permuting.

References

[1] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

[2] Laurence Aitchison, Adam Yang, and Sebastian W Ober. Deep kernel processes. In *International Conference on Machine Learning*, pages 130–140. PMLR, 2021.

[3] Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. *arXiv preprint arXiv:2201.12250*, 2022.

[4] 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.

[5] Olivier Collier and Arnak S Dalalyan. Minimax rates in permutation estimation for feature matching. *The Journal of Machine Learning Research*, 17(1):162–192, 2016.

[6] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. *Advances in neural information processing systems*, 28, 2015.

[7] 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.

[8] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *ICLR*, 2021.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[10] 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.

[11] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[12] 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.

[13] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[15] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[18] Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, page 4, 2008.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[20] Aarre Laakso and Garrison Cottrell. Content and cluster analysis: assessing representational similarity in neural systems. *Philosophical psychology*, 13(1):47–76, 2000.

[21] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[22] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *Feature Extraction: Modern Questions and Challenges*, pages 196–212. PMLR, 2015.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[24] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[25] Kristof Meding, Luca M Schulze Buschoff, Robert Geirhos, and Felix A Wichmann. Trivial or impossible–dichotomous data difficulty masks model differences (on imagenet and beyond). *arXiv preprint arXiv:2110.05922*, 2021.

[26] Johannes Mehrer, Courtney J Spoerer, Nikolaus Kriegeskorte, and Tim C Kietzmann. Individual differences among deep neural network models. *Nature communications*, 11(1):1–12, 2020.

[27] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

[28] Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[30] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

[31] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ICLR 2014*.

[32] 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.

[33] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[34] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13699–13708, 2022.

[35] 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.

[36] Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. *Advances in neural information processing systems*, 31, 2018.

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[38] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.