The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks

Anonymous Authors

August 5, 2021

Abstract

In this paper, we conjecture that if the permutation invariance of neural networks is taken into account, SGD solutions will likely have no barrier in the linear interpolation between them. Although a bold conjecture, we show how extensive empirical attempts fall short of refuting it. We further provide a theoretical result to support our conjecture. Our conjecture has implications for lottery ticket hypothesis, distributed training and ensemble methods. Our source code is available at https://github.com/Neurips21Permutation/PermutationInvariance.

1 Introduction

Understanding the loss landscape of deep neural networks has been the subject of many studies due to its close connections to optimization and generalization (Li et al., 2017; Mei et al., 2018; Geiger et al., 2019; Nguyen et al., 2018; Fort et al., 2019; Baldassi et al., 2020). Empirical observations suggest that loss landscape of deep networks has many minima (Keskar et al., 2017; Draxler et al., 2018; Zhang et al., 2017). One reason behind the abundance of minima is over-parametrization. Over-parametrized networks have enough capacity to present different functions that behave similarly on the training data but vastly different on other inputs (Neyshabur et al., 2017; Nguyen et al., 2018; Li et al., 2018; Liu et al., 2020). Another contributing factor is the existence of scale and permutation invariances which allows the same function to be represented with many different parameter values of the same network and imposes a counter-intuitive geometry on the loss landscape (Neyshabur et al., 2015; Brea et al., 2019a).

Previous works study the relationship between different minima found by SGD and establish that they are connected by a path of non-increasing loss; however, they are *not* connected by a *linear* path (Freeman and Bruna, 2016; Draxler et al., 2018; Garipov et al., 2018). This phenomenon is often referred to as mode connectivity (Garipov et al., 2018) and the loss increase on the path between two solutions is often referred to as (energy) barrier (Draxler et al., 2018). Understanding *linear mode connectivity* (LMC) is highly motivated by several direct conceptual and practical implications from pruning and sparse training to distributed optimization and ensemble methods.

The relationship between LMC and pruning was established by Frankle et al. (2020) where they showed the correspondence between LMC and the well-known lottery ticket hypothesis (LTH) (Frankle and Carbin, 2019). In short, LTH conjectures that neural networks contain sparse subnetworks that can be trained in isolation, from initialization, or early in training to achieve comparable test accuracy. Frankle et al. (2020) showed that solutions that are linearly connected with no barrier have the same lottery ticket. They further discuss how linear-connectivity is associated with stability of SGD. This view suggests that SGD solutions that are linearly connected with no barrier can be thought of as being in the same basin of the loss landscape and once SGD converges to a basin, it shows a stable behavior inside the basin[1]. Because of the direct correspondence between LMC and LTH, any understanding of LMC, has implications for LTH, stability of SGD and pruning techniques.

[1]This notion of basin is also consistent with the definition proposed by Neyshabur et al. (2020).

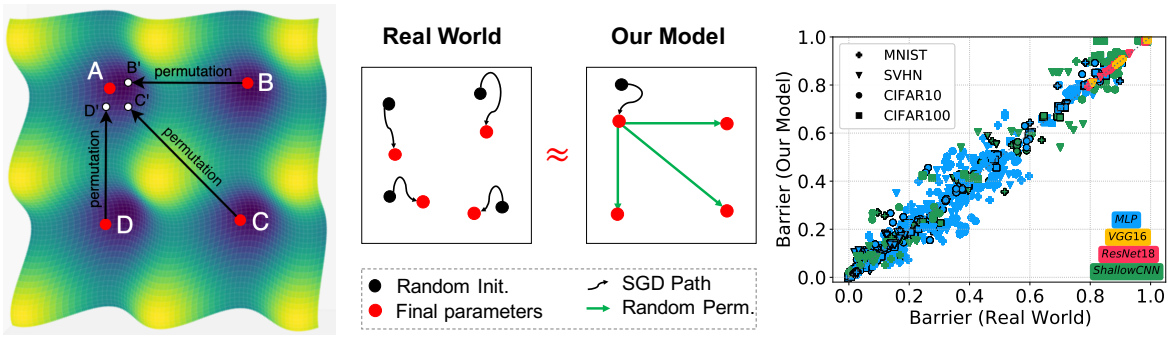


Figure 1: **Linear mode connectivity when using permutation invariance**. **Left:** Schematic picture of four minima A, B, C, D in different basins with an energy barrier between each pair. However, our conjecture suggests that permuting hidden units of B, C and D would result in B', C' and D' which present the exact same function while having no barrier on their linear interpolation with A. **Middle:** Our model for *barriers* in real world SGD solutions. In real world we train networks by running SGD with different random seeds starting from different initializations. In our model, different final networks are achieved by applying random permutations to the same SGD solution (or equivalently, applying random permutations to the same initialization and then running SGD with the same seed on them). **Right:** Aggregation of our extensive empirical evidence (more than 3000 trained networks) in one plot comparing barriers in real world against our model across different choices of architecture family, dataset, width, depth, and random seed. Poits with solid edges correspond to lowest barrier found after searching in the space of valid permutations using a Simulated Annealing (SA). SA shows better performance for shallow networks pushing more solid edge points to the lower left corner.

Linear mode connectivity has also direct implications for ensemble methods and distributed training. Ensemble methods highly depend on an understanding of the loss landscape and being able to sample from solutions. Better understanding of mode connectivity has been shown to be essential in devising better ensemble methods (Garipov et al., 2018). Linear mode connectivity between solutions or checkpoints also allows for weight averaging techniques for distributed optimization to be used as effectively in deep learning as convex optimization (Scaman et al., 2019).

In this paper, we conjecture that by taking permutation invariance into account, the loss landscape can be simplified significantly resulting in linear mode connectivity between SGD solutions. We investigate this conjecture both theoretically and empirically through extensive experiments. We show how our attempts fall short of refuting this hypothesis and end up as supporting evidence for it (see Figure 1). We believe our conjecture sheds light into the structure of loss landscape and could lead to practical implications for the aforementioned areas.

Contributions. This paper makes the following contributions:

- We study linear mode connectivity (LMC) between solutions trained from different initializations and investigate how it is affected by choices such as width, depth and task difficulty for fully connected and convolutional networks (Section 2).
- We introduce our main conjecture in Section 3: If invariances are taken into account there will likely be no barrier on the linear interpolation of SGD solutions (see the left panel of Figure 1).
- By investigating the conjecture theoretically, we prove that it holds for a wide enough fully connected network with one hidden layer at random initialization (Section 3).
- In Section 4, we provide strong empirical evidence in support of our conjecture. To overcome the computational challenge of directly evaluating the hypothesis empirically which requires searching in the space of all possible permutations, we propose an alternative approach. We consider a set of solutions corresponding to random permutations of a single fixed SGD solution (our model) and show several empirical evidences suggesting our model is a good approximation for all SGD solutions(real world) with different random seeds (see the middle and right panel of Figure 1).

Further related work. Permutation symmetry of neurons in every layer results in multiple equivalent minima connected via saddle points. Few studies investigate the role of these symmetries in the context of connectivity of different basins. Fukumizu and Amari (2000) prove that a point corresponding to the global minimum of a smaller model can be a local minimum or a saddle point of the larger model. Brea et al. (2019b) find smooth paths between equivalent global minima that lead through a permutation point, *i.e.*, where the input and output weight vectors of two neurons in the same hidden layer interchange. They describe a method to permute all neuron indices in the same layer at the same cost. Tatro et al. (2020) showed that aligning the neurons in two different neural networks makes it easier to find second order curves between them in the loss landscape where barriers are absent.

2 Loss Barriers

In this section we first give a formal definition for linear mode connectivity and then study how it is affected by different factors such as network width, depth, and task difficulty for variety of architecture.

2.1 Definitions

Let $f_\theta(\cdot)$ be a function presented by a neural network with parameter vector θ that includes all parameters and $\mathcal{L}(\theta)$ be the any given loss (e.g. train or test error) of $f_\theta(\cdot)$. Let $\mathcal{E}_\alpha(\theta_1, \theta_2) = \mathcal{L}(\alpha\theta_1 + (1-\alpha)\theta_2)$, for $\alpha \in [0, 1]$ be the loss of the network created by linearly interpolating between parameters of two networks $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$. The loss barrier $B(\theta_1, \theta_2)$ along the linear path between θ_1 and θ_2 is defined as the highest difference between the loss occurred when linearly connecting two points θ_1, θ_2 and linear interpolation of the loss values at each of them:

$$B(\theta_1, \theta_2) = \sup_\alpha \mathcal{E}_\alpha[\mathcal{L}(\alpha\theta_1 + (1-\alpha)\theta_2)] - [\alpha\mathcal{L}(\theta_1) + (1-\alpha)\mathcal{L}(\theta_2)]. \tag{1}$$

The above definition differs from what was proposed by Frankle et al. (2020) in that they used $0.5\mathcal{L}(\theta_1) + 0.5\mathcal{L}(\theta_2)$ instead of $\alpha\mathcal{L}(\theta_1) + (1-\alpha)\mathcal{L}(\theta_2)$ in our definition. These definitions are the same when $\mathcal{L}(\theta_1) = \mathcal{L}(\theta_2)$ but when they differ, we find our definition to be more appropriate because our definition would assign no barrier value to a loss that is changing linearly between θ_1 and θ_2.

We say that two networks θ_1 and θ_2 are linear mode connected if the barrier between them along a linear path is ≈ 0 (Frankle et al., 2020). It has been observed in the literature that any two minimizers of a deep network can be connected via a non-linear low-loss path (Garipov et al., 2018; Draxler et al., 2018; Fort and Jastrzebski, 2019). This work examines *linear* mode connectivity (LMC) between minima. Next, we empirically investigate the affect of task difficulty and choices such as architecture family, width and depth on LMC of SGD solutions.

2.2 Empirical Investigation: Barriers

In this section, we look into barriers between different SGD solutions on all combinations of four architecture families (MLP (Rosenblatt, 1961), Shallow-CNN (Neyshabur, 2020), ResNet (He et al., 2015) and VGG (Simonyan and Zisserman, 2015)) and four datasets (MNIST (LeCun and Cortes, 2010), SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009)). We now empirically investigate how different factors such as architecture family, width, depth and task difficulty impact the barrier size. [2]. In the following section we refer to training loss barrier as barrier. For loss barriers on test set see C.2 . For train and test error see A.2 .

Width. We evaluate the impact of width on the barrier size in Figure 2. We note that for large values of width the barrier becomes small. This effect starts at lower width for simpler datasets such as MNIST

[2]In all plots the barrier is evaluated for across 5 different random pairs (10 random SGD solutions). In our experiments, we have observed that evaluating the barrier at $\alpha = \frac{1}{2}$ is a reasonable surrogate for taking the supremum over α (the difference is less than 10^{-4}). Therefore, to save computation, we report the barrier value at $\alpha = \frac{1}{2}$.

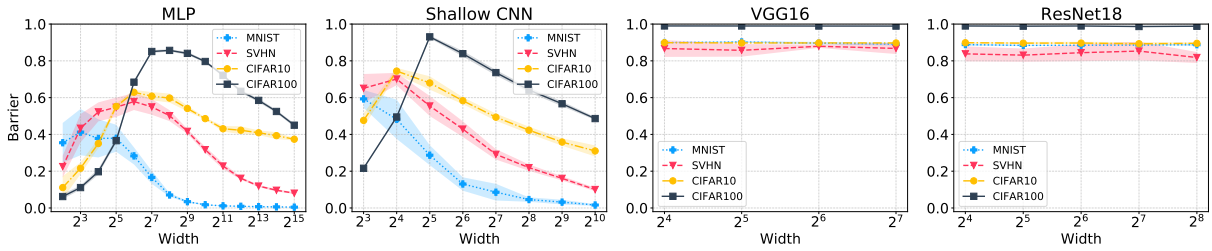


Figure 2: **Effect of width on barrier size.** From **left to right**: one-layer MLP, two-layer Shallow-CNN, VGG-16 and ResNet-18 architectures and MNIST, CIFAR-10, SVHN, CIFAR-100 datasets. For large width sizes the barrier becomes small. This effect starts at lower width for simpler datasets such as MNIST and SVHN compared to CIFAR datasets. Taking a closer look shows a similar trend to that of double-descent phenomena. MLP architectures hit their peak at a lower width compared to CNNs and decreasing trend starts earlier. For ResNet, the barrier size is saturated at a high value and does not change due to the effect of depth as discussed in Figure 3.

and SVHN compared to CIFAR datasets. Taking a closer look we notice that the barrier increases with width up to a point and beyond that increasing width leads to lower barrier size. This effect is reminiscent of the double descent phenomena (Belkin et al., 2019) though checking the test error indicates that in our experiments the peak happens at slightly larger size that needed to fit the training data. This phenomena is observed for both fully connected and convolutional architectures. MLP architectures hit their peak at a lower width compared to CNNs and decreasing trend starts earlier. For ResNets the barrier size is saturated at a high value and does not change. We removed the lines corresponding to VGG architecture on different datasets and the reason is that, similar to ResNets, their barrier value is saturated at a high value and does not change by increasing the width. The behavior observed for both ResNets and VGG architectures is due to the effect of depth as discussed in the next paragraph and depicted in Figure 3.

Depth. We varied network depth in Figure 3 to evaluate its impact on the barrier between optimal solutions obtained from different initializations. For MLPs, we fixed the layer width at 2^{10} while adding identical layers as shown along the x-axis. We observe a fast and significant barrier increase as more layers are added. For VGG architecture family we observe significant barrier. This might be due to the convolutions itself or the effect of depth. In order to shed light on this observation, we use Shallow-CNN (Neyshabur, 2020) with only two convolutional layers. As can be seen in Figure 3 when Shallow-CNN has two layers the barrier size is low, while keeping the layer width fixed at 2^{10} and adding more layers increases the barrier size. For residual networks we also consider three ResNet architectures with 18, 34 and 50 layers and we observe the same barrier sizes as VGG for all these depth values. The main overall observation from depth experiments is that observe that for both fully connected and convolutional architectures, increasing depth increases the barrier size significantly so the effect of depth is not similar to width. This can also be attributed to the observations that deeper networks usually have less smooth landscape (Li et al., 2017).

Task difficulty and architecture choice. In Figure 4 we look into the impact of the task difficulty provided by the dataset choice (MNIST, SVHN, CIFAR-10, CIFAR-100, and ImageNet (Deng et al., 2009)) and the architecture type (one-layer MLP with 2^{12} neurons, Shallow-CNN with two convolutional layer and width of 2^{10} , VGG-16 with batch-normalization, ResNet18 and ResNet50). Each row in Figure 4a and Figure 4b shows the effect of task difficulty *e.g.*, fixing the task to SVHN and moving from MLP to Shallow-CNN gives lower test error hence lower barrier size. Each column also represents the effect of architecture on a specific dataset *e.g.*, fixing the architecture to Shallow-CNN and moving from CIFAR10 to CIFAR100 presents an increase in test error, hence increase in the barrier size. Figure 4c aggregates the correlation between test error and size of the barrier. Although deep architectures like VGG16 and ResNet18 presents low test error, they form a cluster in the top-left. The discussed effect of depth overweights the effect of generalization here, as VGG16 and ResNets have high barrier for different datasets. For MLP and Shallow-CNN we observe a high positive correlation between test error and barrier size across different datasets.

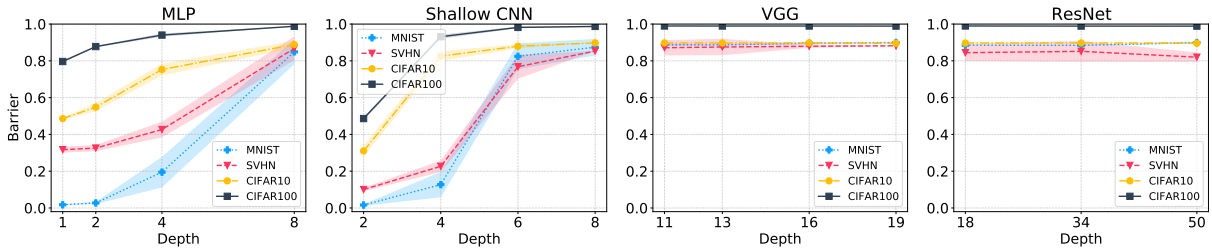
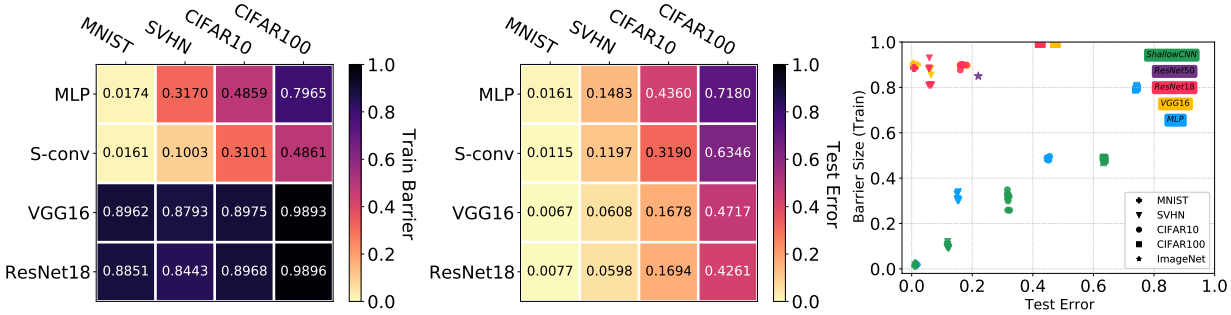


Figure 3: **Effect of depth on barrier size.** From **left to right** one-layer MLP, two-layer Shallow-CNN, VGG(11,13,16,19), and ResNet(18,34,50) architectures and MNIST, CIFAR-10, SVHN, CIFAR-100 datasets. For MLP and Shallow-CNN, we fixed the layer width at 2^{10} while adding identical layers as shown along the x-axis. Similar behavior is observed for fully connected and CNN family, *i.e.*, low barrier when number of layers are low while we observe a fast and significant barrier increase as more layers are added. Increasing depth leads to higher barrier values until it saturates (as seen for VGG and ResNet).



(a) Barrier for different architectures and task difficulty

(b) Achieved test error

(c) Lower test error results in a lower barrier for shallow networks

Figure 4: **Effect of architecture choice and task difficulty on barrier size.** Each row in Figure 4a and Figure 4b shows the effect of task difficulty while each column represents the effect of architecture on a specific dataset. Figure 4c notes that a pair of (architecture, task) has lower barrier if the test error is lower. Therefore, any changes in the architecture or the task that improves the test error, also improves the loss barrier. Effect of depth is stronger than (architecture, task) which leads to high barrier values for ResNets on MNIST, SVHN, CIFAR10, CIFAR100, and ImageNet.

3 Role of Invariance in Loss Barriers

Understanding the loss landscape of deep networks has proven to be very challenging. One of the main challenges in studying the loss landscape without taking the optimization algorithm into account is that there exist many minima with different generalization properties. Most of such minima are not reachable by SGD and we only know about their existence through artificially-made optimization algorithms and training regimes (Neyshabur et al., 2017). To circumvent this issue, we focus on parts of the landscape that are reachable by SGD. Given a dataset and an architecture, one could define a probability distribution over all solutions reached by SGD and focus on the subset where SGD is more likely to converge to.

3.1 Invariances in neural network function class

We say that a network is *invariant* with respect to a transformation if and only if the network resulted from the transformation represents the same function as the original network. There are two well-known invariances: one is the unit-rescaling due to positive homogeneity of ReLU activations (Neyshabur et al., 2015) and the other is permutation of the hidden units. Unit-rescaling has been well-studied and empirical

evidence suggests that implicit bias of SGD would make the solution converge to a stage where the weights are more balanced (Neyshabur et al., 2015; Wu et al., 2019). Since we are interested in the loss landscape through the lens of SGD and SGD is much more likely to converge to a particular rescaling, consideration of this type of invariance does not seem useful. However, in the case of permutations, all permutations are equally likely for SGD and therefore, it is important to understand their role in the geometric properties of the landscape and its basins of attraction.

Here we consider invariances that are in form of permutations of hidden units in each layer of the network, *i.e.*, each layer i with parameters W_i is replaced with $P_i W_i P_{i-1}$ where P_i is a permutation matrix and $P_l = P_0$ is the identity matrix. Note that our results only hold for permutation matrices since only permutation commutes with the nonlinearity. We use $\mathcal{P}$ to refer to the set of valid permutations for a neural network and use π to refer to a valid permutation.

3.2 Our Conjecture

As mentioned above, SGD's implicit regularization balances weight norms and therefore, scale invariance is not an issue in neural networks. Here we focus on permutation invariance. We first state our conjecture informally:

Most SGD solutions belong to a set $\mathcal{S}$ whose elements can be permuted in such a way that there is no barrier on the linear interpolation between any two elements.

The above conjecture suggests most SGD solutions end up in the same basin in the loss landscape after proper permutation(see Figure 1 left pane). We recognize that the above This has great practical implications for model ensembling and parallelism since one can average models that are in the same basin in the loss landscape.

The above statement can be formalized as follows:

Conjecture 1 (Main Conjecture)**.** *Let $f(\theta)$ be the function representing a neural network with parameters $\theta \in \mathbb{R}^k$, $\mathcal{P}$ be the set of all valid permutations for the network, $P : \mathbb{R}^k \times \mathcal{P} \to \mathbb{R}^k$ be the function that applies a given permutation to parameters and returns the permuted version, and $B(\cdot, \cdot)$ be the function that returns barrier value between two solutions as defined in Equation 1. Then, there exists $h > 0$ such that for any network $f(\theta)$ of minimum width at least h the following holds: There exist a set of solutions $\mathcal{S} \subseteq \mathbb{R}^k$ and a function $Q : \mathcal{S} \to \mathcal{P}$ such for any $\theta_1, \theta_2 \in \mathcal{S}$, $B(P(\theta_1, Q(\theta_1)), \theta_2) \approx 0$ and with high probability over an SGD solution θ, $\theta \in \mathcal{S}$.*

Next, we approach Conjecture 1 from both theoretical and empirical aspects and provide evidence to support it.

3.3 Theoretical Result

Providing theoretical results for neural network models is difficult due to complexity of function class and hence, theoretical investigations usually cover simpler cases. Nevertheless, theoretical investigations help in providing intuition into why/when a phenomena occurs. Here, for example, we provide theoretical reasoning into the phenomena proposed in Conjecture 1 when network size is captured by network width; when network width becomes larger than a threshold, it is possible to find a permutation on hidden units such that it leads to having no barrier.

Here, we formally state the theorem.

Theorem 3.1. *Let h be the number of hidden units, d be the input size. Let the function $f_{\mathbf{v},\mathbf{U}}(\mathbf{x}) = \mathbf{v}^\top \sigma(\mathbf{U}\mathbf{x})$ where $\sigma(\cdot)$ is ReLU activation, $\mathbf{v} \in \mathbb{R}^h$ and $\mathbf{U} \in \mathbb{R}^{h \times d}$ are parameters and $\mathbf{x} \in \mathbb{R}^d$ is the input. We show that if each element of $\mathbf{U}$ and $\mathbf{U}'$ is sampled uniformly from $[-1/\sqrt{d}, 1/\sqrt{d}]$ and each element of $\mathbf{v}$ and $\mathbf{v}'$ is sampled uniformly from $[-1/\sqrt{h}, 1/\sqrt{h}]$, then for any $\mathbf{x} \in \mathbb{R}^d$ such that $\|\mathbf{x}\|_2 = \sqrt{d}$, with probability $1 - \delta$ over $\mathbf{U}, \mathbf{U}', \mathbf{v}, \mathbf{v}'$, there exist a permutation such that*

$$\left| f_{\alpha\mathbf{v}+(1-\alpha)\mathbf{v}'', \alpha\mathbf{U}+(1-\alpha)\mathbf{U}''}(\mathbf{x}) - \alpha f_{\mathbf{v},\mathbf{U}}(\mathbf{x}) - (1-\alpha) f_{\mathbf{v}',\mathbf{U}'}(\mathbf{x}) \right| = \tilde{O}(h^{-\frac{1}{2d+4}})$$

where $\mathbf{v}''$ *and* $\mathbf{U}''$ *are permuted versions of* $\mathbf{v}'$ *and* $\mathbf{U}'$.

Proof is given in the Supplementary Section B.

Theorem 3.1 states that for networks bigger than a size (captured in terms of width) that depends on input dimension one can find a permutation that leads to having no barrier. This statement covers cases of randomly initialized networks. We know that if we start with two random initializations that are close, performing SGD on parameters keeps them close. Therefore, if one finds a permutation that brings two random initialization close to each other, for wide enough networks, when we run SGD on their parameters they stay close.

Theorem 3.1 serves as the first step in proving the Conjecture.

3.4 Empirical approaches to Conjecture 1

Another possible approach is to use brute-force (BF) search mechanism and find the function Q for elements of $\mathcal{S}$. The factorial growth of the number of permutations with the size of hidden units in each layer hinders exhaustive search for a winning permutation π to linear mode connect $P(\theta_1, \pi)$ and θ_2. Even for MLPs with just one hidden layer brute-force works in reasonable time up to 2^4 neurons only, forcing the search to examine $2^4! \approx 2 \cdot 10^{13}$ permuted networks. BF is not feasible even for modest size deep networks. For small networks, one can use BF to find permutations between different models (see Appendix C.1). However, small size networks are not the focus of this paper and Conjecture 1 specifically mentions that.

Given the size of search space, using a more advanced search algorithm can be useful. The issue with this approach is that since it relies on the strength of search algorithm, if the search algorithm fails in finding the permutation, one cannot be sure about the source of failure being the search algorithm or nonexistence of a permutation that leads to no barrier.

3.5 Our model

We propose the following mode as an approach to circumvent the above obstacles. We create a competing set $\mathcal{S}'$ (our model) as a proxy for set $\mathcal{S}$ (real world). Given a solution $\theta_1 \in \mathcal{S}$, we define $\mathcal{S}' = \{P(\theta_1, \pi) | \forall \pi \in \mathcal{P}\}$. We know that set $\mathcal{S}'$ satisfies the conjecture. For set $\mathcal{S}'$, all points are known permutations of θ_1. Therefore, one can permute all points to remove their barriers with θ_1, i.e, for all $\theta_2 = P(\theta_1, \pi)$, one can use $Q(\theta_2) = \pi^{-1}$ to remove the barrier between θ_1, θ_2. We effectively want to show that $\mathcal{S} \sim \mathcal{S}'$ in terms of barrier behavior. Equivalence of $\mathcal{S}'$ and $\mathcal{S}$ in terms of barriers, means that if we choose one element of $\mathcal{S}$ (let's call it θ), one should be able to find permutations for each of other elements of the $\mathcal{S}$ so that the permuted elements have no barrier with θ and hence are in the same basin as θ. The consequence of the equivalence of our model to real world is that Conjecture 1 holds. The conjecture effectively means that different basins exist because of the permutation invariance and if permutation invariance is taken into account (by permuting solutions to remove the barriers between them), there is only one basin, i.e., all solutions reside in the same basin in the loss landscape. We actually want to show $\mathcal{S} \sim \mathcal{S}'$ in terms of optimizing over all permutations but that is not possible so we show $\mathcal{S} \sim \mathcal{S}'$ in different but similar ways to the brute-force search. This would allow us to accumulate evidence for the hypothesis.

4 Empirical Investigation

In this section we show that $\mathcal{S}$ and $\mathcal{S}'$ have similar loss barrier along different factors such as width, depth, architecture, dataset and other model parameters. We first introduce the search algorithms used to find the winning permutation π which removes the barrier to overcome the BF limitation. Then, we analyse the performance of these algorithms on $\mathcal{S}$ and $\mathcal{S}'$ to show similarities in barrier changes are impossible to ignore, even for the case where $\mathcal{S}$ and $\mathcal{S}'$ are chosen minimal.

Algorithm 1 Simulated Annealing (SA) for Permutation Search

procedure SA($\{\theta_i\}, i = 1..n, n \geq 2$) ▷ Goal: minimize the barrier between n solutions
  $\pi_i = \pi_0, \forall i = 1..n$
  for $k = 0$; $k < k_{\max}$; $k{+}{+}$ **do**
    $T \leftarrow \text{temperature}(\frac{k+1}{k_{\max}})$
    Pick random candidate permutations $\{\hat{\pi}_i\}, \forall i = 1..n$
    if $\Psi(P(\theta_i, \hat{\pi}_i)) < \Psi(P(\theta_i, \pi_i))$ **then**
      $\pi_i \leftarrow \hat{\pi}_i$
  return $\{\pi_i\}$

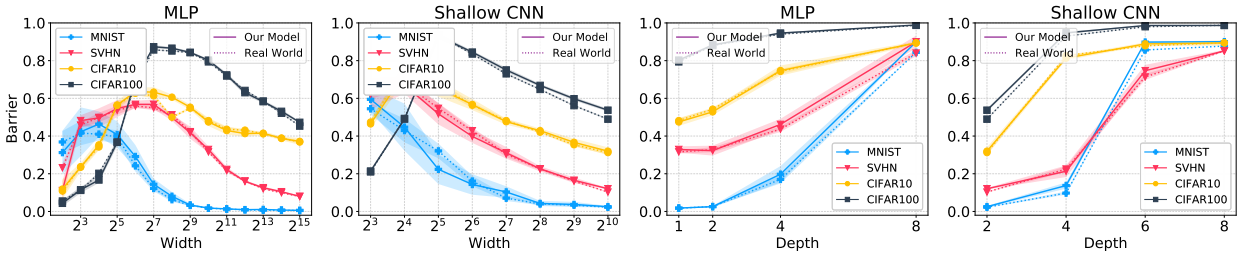


Figure 5: **Similar loss barrier between real world and our model *before* applying permutation.** Increasing width first increases and then decreases the barrier, while adding more layers significantly increases the barrier size. Our model and real world are aligned along different architectures and datasets.

4.1 Search algorithms

The problem of finding a winning permutation $\pi \in \mathcal{P}$ is a variant of the travelling salesman problem (TSP) where neurons are mapped to cities visited by a salesman. The problem belongs to the class of NP-hard optimization problems and *simulated annealing* (SA) is often used to find a solution. SA's performance however highly depends on the parameter choice, including the minimum and maximum temperatures, the cooling schedule and the number of optimization steps. The pseudo code of SA is shown in Algorithm 7. SA takes a set of solutions $\{\theta_i\}, i = 1..n, n \geq 2$ as input (we use $n = 5$) and searches for a set of permutations $\{\pi_i\}$ that reduce the barrier between all permuted $\frac{n(n-1)}{2}$ solution pairs. To find the best $\{\pi_i\}$, in each step of SA the current candidate permutations $\{\hat{\pi}_i\}, i = 1..n$ are evaluated to minimize the function Ψ. We use several version of simulated annealing to solve the conjecture. These vary in their definition of Ψ. In the first version SA_1, we evaluate the average pairwise barrier between candidate permutations $B(P(\theta_i, \pi_i), P(\theta_j, \pi_j)), i \neq j$. In SA_2, we average the weights of permuted models $P(\theta_i, \pi_i)$ first and then compute the train error of the resulting average model. These two versions try to find $\{\pi_i\}$ such that the permuted models reside in one basin hence achieving zero barrier. Our empirical results suggest that SA_1 and SA_2 yield very similar performance. However, SA_2 is significantly less computationally expensive, which makes it better suitable to explore larger models. In the following sections we present the results obtained with SA_2 only and refer to this version as SA.

4.2 Similarity of $\mathcal{S}$ and $\mathcal{S}'$

In this Section, we show that $\mathcal{S}$ and $\mathcal{S}'$ have similar barrier in terms of loss by showing the effect of changing different architecture parameters such as as width, depth across various datasets. Figure 5 shows the effect of width and depth on barrier sizes for MLP and Shallow-CNN across different datasets. Here we consider the mean over all pair-wise barriers as barrier size. As discussed in Section 2.2 increasing width first increases and then decreases the barrier, while adding more layers significantly increases the barrier size. In each plot, barrier size for $\mathcal{S}$ is aligned with $\mathcal{S}'$. Such alignment is also observed in Figure 6, where we see the barrier after permutation using SA. These two observations confirm the similarity between $\mathcal{S}$ and $\mathcal{S}'$. Comparing

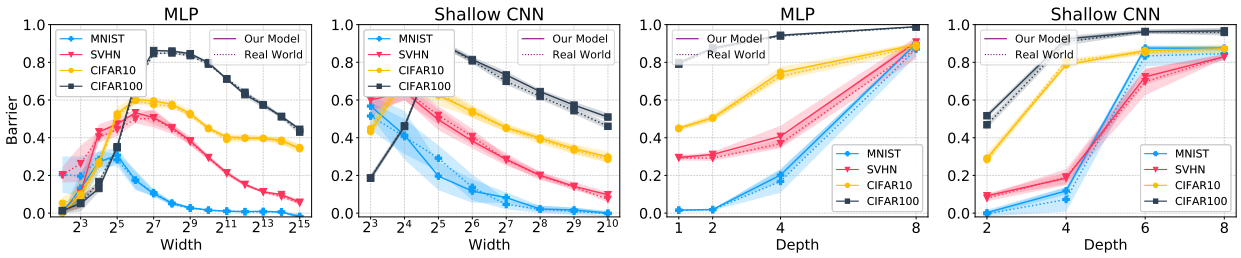


Figure 6: **Similar loss barrier between real world and our model *after* applying permutation.** Effects of width and depth also holds in this setting. Compared to Figure 5, we observe slight barrier reduction. Reducing search space helps SA to find better solutions (see section 4.3).

Figure 5 and Figure 6 also shows that SA is not able to find permutations to apply for each SGD solution, such that the pair-wise barrier is removed. We know that SA does not guarantee to find a solution and is known to lose its effectiveness on TSP benchmarks beyond 1'000 cities (Zhan et al., 2016). The effectiveness of SA is also reduced here as we can only evaluate the cost of full path. One way to increase this effectiveness is to reduce the search space. We will discuss this intervention in details in the next section.

4.3 Search space reduction

In order to reduce the search space, here we only take two SGD solutions θ_1 and θ_2, permute θ_1 and report the barrier between permuted θ_1 and θ_2 as found by SA with $n = 2$. Figure 7 and Figure 8 show the effect of width and depth on barrier similarity between $\mathcal{S}$ and $\mathcal{S}'$ before and after permutation. We note SA success on barrier removal by comparing Figure 7 and Figure 8. SA performance on $\mathcal{S}$ and $\mathcal{S}'$ yields similar results for both before and after permutation scenarios. Such similar performance is observed along a wide range of width and depth for both MLP and Shallow-CNN over different datasets(MNIST, SVHN, CIFAR10, CIFAR100). We look into effects of changing model size in terms of width and depth as in earlier Sections, and note that similar trends hold for before and after permuting solution θ_1. Comparing Figure 7 and Figure 8 shows that reducing the search space makes SA more successful in finding the permutation $\{\pi\}$ to remove the barriers. Specifically, SA could indeed find permutations across different networks and datasets that when applied to θ_1 result in zero barrier. SA could also find permutations that reduce the barrier for both MLP and Shallow-CNN across different width, depth and datasets. For example, such cases include MLP for MNIST, SVHN, CIFAR10, and CIFAR100 where depth is 1 and width is 2^3 and 2^4, MLP for MNIST where depth is 2 and width is 2^{10}, Shallow-CNN for MNIST, SVHN, CIFAR10, CIFAR100 where depth is 2 and width is 2^4 and for Shallow-CNN for MNIST where depth is 2 and width is 2^6.

Empirical investigations in Sections 4.2, 4.3 provide several pieces of evidence that $\mathcal{S}, \mathcal{S}'$ show similar barrier behavior across a variety of architecture and dataset choices. Moreover, when we reduce the search space, SA is successful in removing barrier between pairs of elements in $\mathcal{S}, \mathcal{S}'$ which suggest another evidence that, one can remove loss barriers between elements of both sets. Putting together, they support our main conjecture that one can remove loss barriers between different minimas on the neural network.

5 Discussions and Conclusion

We investigated the loss landscape of ReLu networks and proposed and probed the conjecture that the barriers in the loss landscape between different solutions of a neural network optimization problem are an artifact of ignoring the permutation invariance of the function class. In a nutshell, if one considers permutation invariance, there is essentially no loss barrier between different solutions and they all exist in the same basin in the loss landscape.

Our analysis has direct implication on initialization schemes for neural networks. Essentially it postulates that randomness in terms of permutation does not impact the quality of the final result. It is interesting to

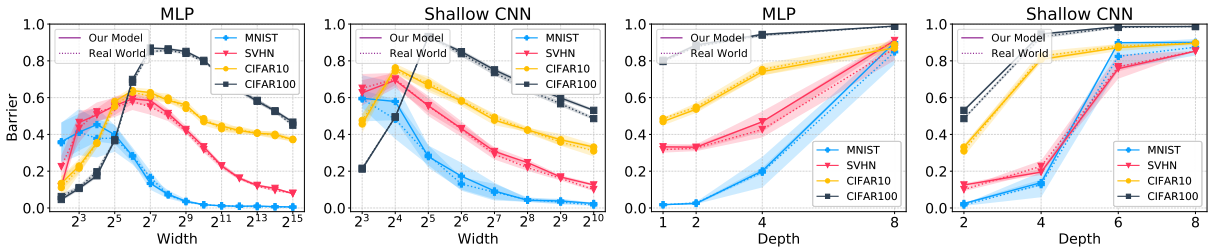


Figure 7: **Effect of width and depth on barrier consistency between real world and our model *before* permutation**. Search space is reduced here *i.e.*, we take two SGD solutions θ_1 and θ_2, permute θ_1 and report the barrier between permuted θ_1 and θ_2 as found by SA with $n = 2$. Similarity of loss barrier between real world and our model is preserved across model type and dataset choices as width and depth of the models are increased.

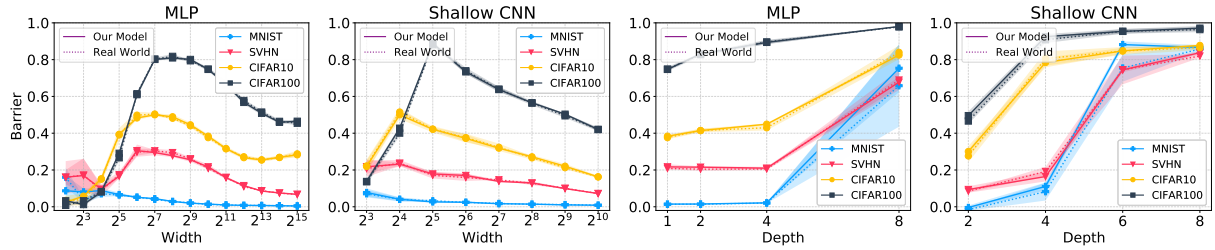


Figure 8: **Effect of width and depth on barrier consistency between real world and our model *after* permutation.** We observe that reducing the search space makes SA more successful in finding the permutation $\{\pi\}$ to remove the barriers. Specifically, SA could indeed find permutations across different networks and datasets that when applied to θ_1 result in zero barrier *e.g.*, MLP across all datasets where depth is 1 and width is 2^3 and 2^4, MLP for MNIST where depth is 2 and width is 2^{10}, Shallow-CNN for MNIST, SVHN, CIFAR10, CIFAR100 where depth is 2 and width is 2^4.

explore whether it is possible to come up with initialization that does not have permutation invariance and only acts like a perturbation to the same permutation. If all basins in the loss landscape are basically the same function, there will be no need to search all of them. One can explore the same basin while looking for diverse solutions and this makes search much easier and would lead to substantially more efficient search algorithms.

Another area where our analysis is of importance is for ensembles and distributed training. If we can track the optimal permutation (that brings all solutions in one basin), it is possible to use it to do weight averaging and ensemble more efficiently. Moreover, we are interested in answering the question whether there is a one-to-one mapping between lottery tickets and permutations. We believe our analysis has provided the first step in investigating these important questions and testing the usefulness of our conjecture in ensemble methods and pruning is the subject of future studies.

The biggest limiting factor of our study is the size of the search space and hence we need a strong search algorithm, specially for deep models where the size and complexity of the search space was prohibitive in terms of computation for the existing search methods. We hope improvements of search algorithms can help us extend our results. Lastly, our analysis focuses on image recognition task and extending this results to natural language tasks is of interest for future work.

References

Baldassi, C., Pittorino, F., and Zecchina, R. (2020). Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1):161–170.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Brea, J., Simsek, B., Illing, B., and Gerstner, W. (2019a). 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*.

Brea, J., Simsek, B., Illing, B., and Gerstner, W. (2019b). Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. (2018). Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR.

Fort, S., Hu, H., and Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

Fort, S. and Jastrzebski, S. (2019). Large scale structure of neural network loss landscapes.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *ICML*, volume 119, pages 3259–3269. PMLR.

Freeman, C. D. and Bruna, J. (2016). Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*.

Fukumizu, K. and Amari, S. (2000). Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13:317–327.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of DNNs.

Geiger, M., Spigler, S., d'Ascoli, S., Sagun, L., Baity-Jesi, M., Biroli, G., and Wyart, M. (2019). Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima.

Krizhevsky, A., Nair, V., and Hinton, G. (2009). Cifar-100 and cifar-10 (canadian institute for advanced research). MIT License.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Creative Commons Attribution-Share Alike 3.0.

Li, D., Ding, T., and Sun, R. (2018). Over-parameterized deep neural networks have no strict local minima for any continuous activations. *arXiv preprint arXiv:1812.11039*.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2017). Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

Liu, C., Zhu, L., and Belkin, M. (2020). Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *arXiv preprint arXiv:2003.00307*.

Mei, S., Montanari, A., and Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. License: CC0: Public Domain.

Neyshabur, B. (2020). Towards learning convolutions from scratch. In *Advances in Neural Information Processing Systems*.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5949–5958.

Neyshabur, B., Salakhutdinov, R., and Srebro, N. (2015). Path-SGD: Path-normalized optimization in deep neural networks. *arXiv preprint arXiv:1506.02617*.

Neyshabur, B., Sedghi, H., and Zhang, C. (2020). What is being transferred in transfer learning? In *NeurIPS*.

Nguyen, Q., Mukkamala, M. C., and Hein, M. (2018). On the loss landscape of a class of deep neural networks with no bad local valleys. *arXiv preprint arXiv:1809.10749*.

Ritchie, S., Slone, A., and Ramasesh, V. (2020). Caliban: Docker-based job manager for reproducible workflows. *Journal of Open Source Software*, 5(53):2403.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY.

Scaman, K., Bach, F., Bubeck, S., Lee, Y., and Massoulié, L. (2019). Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research*, 20:1–31.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*.

Tatro, N. J., Chen, P.-Y., Das, P., Melnyk, I., Sattigeri, P., and Lai, R. (2020). Optimizing mode connectivity via neuron alignment. *arXiv preprint arXiv:2009.02439*.

Wu, X., Dobriban, E., Ren, T., Wu, S., Li, Z., Gunasekar, S., Ward, R., and Liu, Q. (2019). Implicit regularization and convergence for weight normalization. *arXiv preprint arXiv:1911.07956*.

Zhan, S.-h., Lin, J., Zhang, Z.-j., and Zhong, Y.-w. (2016). List-based simulated annealing algorithm for traveling salesman problem. *Intell. Neuroscience*, 2016:8.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations*.

A Implementation details

We used Caliban Ritchie et al. (2020) to manage all experiments in a reproducible environment in Google Cloud's AI Platform. Each point in plots show the mean value taken over 5 different runs.

A.1 Training Hyperparameters

Table 1 summarizes the set of used hyperparameters for training different networks.

Table 1: Training Hyperparameters

Hyper-parameters	MLP	S-conv	VGG	ResNet
Learning Rate	Fixed 0.01[1], Fixed 0.001[2]	Cosine 0.02	Cosine 0.02	Cosine 0.02
Batch Size	64	256	256	256
Epochs	3000	1000	1000	1000
Momentum	0.9	0.9	0.9	0.9
Weight Decay	-	-	-	-
Data Augmentation	Normalization	Normalization	Normalization	Normalization

[1] MNIST
[2] SVHN, CIFAR10, CIFAR100

A.2 Performance Evaluation of Trained Models: Error and Loss

Figure 9 demonstrate Train and Test error/loss for width experiments. Here for MLP we have one hidden layer, for S-conv two convolutional layer, ResNet is fixed to ResNet18, and VGG16 is selected from VGG family. Figure 10 also demonstrate Train and Test error/loss for different depth, setting MLP to have 1024 hidden units in each layer, S-conv , VGG and ResNest to have 1024, 64, 64 channels in each convolutional layer, respectively.

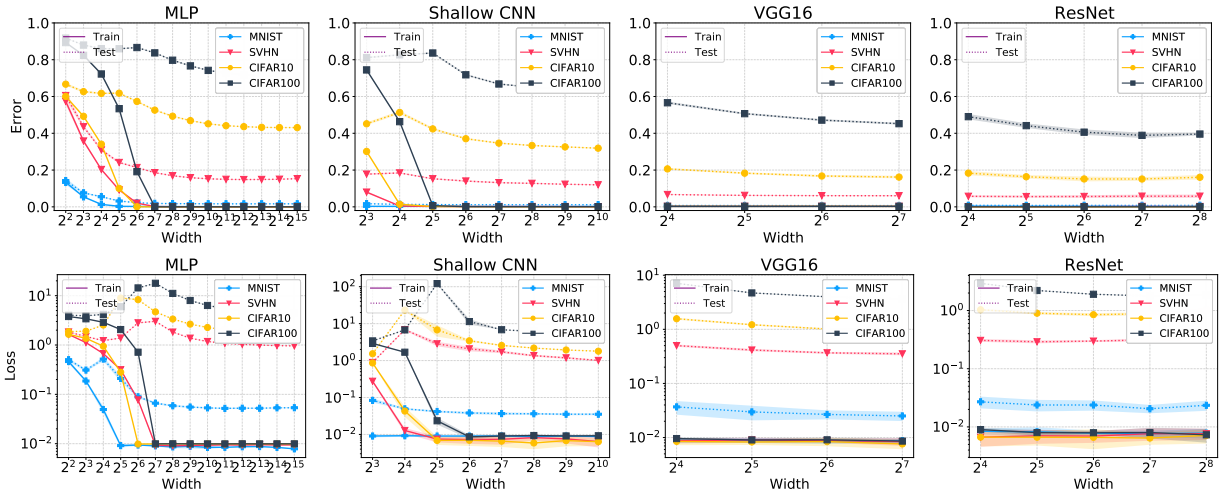


Figure 9: **Train and Test Error/Loss for Width** All the models are trained for 1000 epochs except MLP networks that are trained for 3000 epochs. The stopping criteria is either Cross Entropy Loss reaches 0.01 or number of epochs reaches to maximum epochs

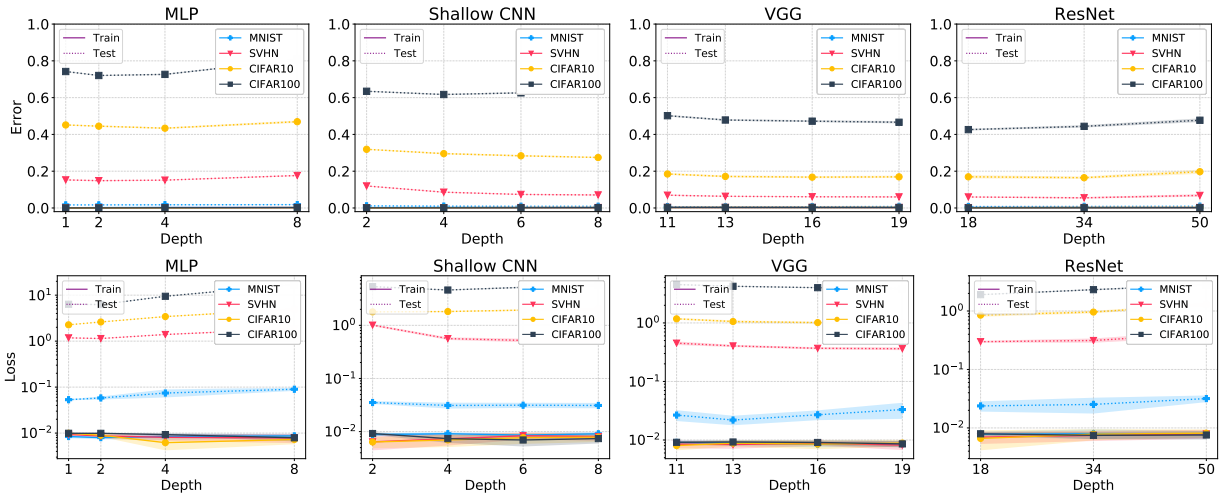


Figure 10: **Train and Test Error/Loss for Depth** All the models are trained for 1000 epochs except MLP networks that are trained for 3000 epochs. The stopping criteria is either Cross Entropy Loss reaches 0.01 or number of epochs reaches to maximum epochs

A.3 Code

We use Simanneal[3] as python module for simulated annealing. The process involves:

- Randomly move or alter the state (generate a permutation)
- Assess the energy of the new state (permuted model) using the objective function (Linear Mode Connectivity based on Equation 1)
- Compare the energy to the previous state and decide whether to accept the new solution or reject it based on the current temperature.

For a move to be accepted, it must meet one of two requirements:

- The move causes a decrease in state energy (i.e. an improvement in the objective function)
- The move increases the state energy (i.e. a slightly worse solution) but is within the bounds of the temperature.

A.3.1 Similarity of $\mathcal{S}$ and $\mathcal{S}'$

The following code runs simulated annealing to find the best permutation in Section 4.2

```
// Simulated Annealing
from simanneal import Annealer
def barrier_SA(arch, model, sd1, sd2, w2, init_state, tmax, tmin, steps, train_inputs, train_targets,
    train_avg_org_models, nchannels, nclasses, nunits):
   class BarrierCalculationProblem(Annealer):
      """annealer with a travelling salesman problem."""
      def __init__(self, state):
         super(BarrierCalculationProblem, self).__init__(state) # important!

      def move(self):
```

[3] https://github.com/perrygeo/simanneal

```
        """Swaps two cities in the route."""
        initial_energy = self.energy()
        for j in range(5):
            for i in range(len(self.state[j])):
                x = self.state[j][i]
                a = random.randint(0, len(x) - 1)
                b = random.randint(0, len(x) - 1)
                self.state[j][i][a], self.state[j][i][b] = self.state[j][i][b], self.state[j][i][a]
        return self.energy() - initial_energy

    def energy(self):
        """Calculates the cost for proposed permutation."""
        permuted_models = []
        for i in range(5):
            permuted_models.append(permute(arch, model, self.state[i], sd2[i], w2[i], nchannels,
                nclasses, nunits))
        #### form one model which is the average of 5 permuted models
        permuted_avg = copy.deepcopy(model)
        new_params = OrderedDict()
        for key in sd2[0].keys():
            param = 0
            for i in range(len(permuted_models)):
                param = param + permuted_models[i][key]
            new_params[key] = param / len(permuted_models)
        permuted_avg.load_state_dict(new_params)
        eval_train = evaluate_model(permuted_avg, train_inputs, train_targets)['top1']
        cost = 1 - eval_train
        return cost

bcp = BarrierCalculationProblem(init_state)
bcp_auto = {'tmax': tmax, 'tmin': tmin, 'steps': steps, 'updates': 100}
bcp.set_schedule(bcp_auto)
winning_state, e = bcp.anneal()
return winning_state
```

B Proof of Theorem 3.1

We first recap Theorem 3.1 below for convenience and then provide the proof

Theorem B.1 (3.1)**.** *Let h be the number of hidden units, d be the input size. Let the function $f_{\mathbf{v},\mathbf{U}}(\mathbf{x}) = \mathbf{v}^\top \sigma(\mathbf{U}\mathbf{x})$ where $\sigma(\cdot)$ is ReLU activation, $\mathbf{v} \in \mathbb{R}^h$ and $\mathbf{U} \in \mathbb{R}^{h\times d}$ are parameters and $\mathbf{x} \in \mathbb{R}^d$ is the input. We show that if each element of $\mathbf{U}$ and $\mathbf{U}'$ is sampled uniformly from $[-1/\sqrt{d}, 1/\sqrt{d}]$ and each element of $\mathbf{v}$ and $\mathbf{v}'$ is sampled uniformly from $[-1/\sqrt{h}, 1/\sqrt{h}]$, then for any $\mathbf{x} \in \mathbb{R}^d$ such that $\|\mathbf{x}\|_2 = \sqrt{d}$, with probability $1-\delta$ over $\mathbf{U}, \mathbf{U}', \mathbf{v}, \mathbf{v}'$, there exist a permutation such that*

$$\left|f_{\alpha\mathbf{v}+(1-\alpha)\mathbf{v}'',\alpha\mathbf{U}+(1-\alpha)\mathbf{U}''}(\mathbf{x}) - \alpha f_{\mathbf{v},\mathbf{U}}(\mathbf{x}) - (1-\alpha) f_{\mathbf{v}',\mathbf{U}'}(\mathbf{x})\right| = \tilde{O}(h^{-\frac{1}{2d+4}})$$

where $\mathbf{v}''$ and $\mathbf{U}''$ are permuted versions of $\mathbf{v}'$ and $\mathbf{U}'$.

Proof. For any given $\xi > 0$, we consider the set $S_\xi = \{-1/\sqrt{d}+\xi, -1/\sqrt{d}+3\xi, \ldots, 1/\sqrt{d}-\xi\}^d$ which has size $(\frac{1}{\xi\sqrt{d}})^d$ [4]. For any $s \in S_\xi$, let $C_s(\mathbf{U})$ be the set of indices of rows of $\mathbf{U}$ that are closest in Euclidean

[4]For simplicity, we assume that ξ is chosen so that $1/\sqrt{d}$ is a multiple of ξ.

distance to s than any other element in S_ξ:

$$C_s(\mathbf{U}) = \{i | s = \arg\min_{s' \in S_\xi} \|\mathbf{u}_i - s'\|_\infty\} \tag{2}$$

where for simplicity we assume that arg min returns a single element. We next use the function C_s to specify a permutation that allows each row in $\mathbf{U}'$ to be close to its corresponding row in $\mathbf{U}$. For every $s \in S_\xi$, we consider a random matching of elements in $C_s(\mathbf{U})$ and $C_s(\mathbf{U}')$ and when the sizes don't match, add the extra items in $\mathbf{U}$ and $\mathbf{U}'$ to the sets I and I' accordingly to deal with them later.

Since each element of $\mathbf{U}$ and $\mathbf{U}'$ is sampled uniformly from $[-1/\sqrt{d}, 1/\sqrt{d}]$, for each row in $\mathbf{U}$ and $\mathbf{U}'$, the probability of being assigned to each $s \in S_\xi$ is a multinomial distribution with equal probability for each s. Given any $s \in S_\xi$, we can use Hoeffding's inequality to bound the size of $|C_s(\mathbf{U})|$ with high probability. For any $t \geq 0$:

$$P\left(||C_s(\mathbf{U})| - (h/|S_\xi|)| \geq t\right) \leq -2\exp(-2t^2/h) \tag{3}$$

By union bound over all rows of $\mathbf{U}$ and $\mathbf{U}'$, with probability $1 - \delta/3$, we have that for every $s \in S_\xi$,

$$\frac{h}{|S_\xi|} - \sqrt{\frac{h}{2}\log(12|S_\xi|/\delta)} \leq |C_s(\mathbf{U})|, |C_s(\mathbf{U}')| \leq \frac{h}{|S_\xi|} + \sqrt{\frac{h}{2}\log(12|S_\xi|/\delta)} \tag{4}$$

Consider I and I' which are the sets of indices that we throw out during the index assignment because of the size mismatch. Then, based on above inequality, we have that with probability $1 - \delta/3$,

$$|I| = |I'| = \frac{1}{2}\sum_{s \in S_\xi} ||C_s(\mathbf{U})| - |C_s(\mathbf{U}')|| \leq |S_\xi|\sqrt{\frac{h}{2}\log(12|S_\xi|/\delta)} \tag{5}$$

We next randomly match the indices in I and I. Let $\mathbf{U}''$ be the matrix after applying the permutation to $\mathbf{U}'$ that corresponds to above matching of rows of $\mathbf{U}'$ to their corresponding row in $\mathbf{U}$. Note that for any $i \in [h] \setminus I$, we have that $\|\mathbf{u}_i - \mathbf{u}''_i\|_\infty \leq 2\xi$ and for $i \in I$, we have $\|\mathbf{u}_i - \mathbf{u}''_i\|_\infty \leq 2/\sqrt{d}$. We next upper bound the left hand side of the inequality in the theorem statement:

$$\begin{aligned}
&\left|f_{\alpha\mathbf{v}+(1-\alpha)\mathbf{v}'',\alpha\mathbf{U}+(1-\alpha)\mathbf{U}''}(\mathbf{x}) - \alpha f_{\mathbf{v},\mathbf{U}}(\mathbf{x}) - (1-\alpha) f_{\mathbf{v}',\mathbf{U}'}(\mathbf{x})\right| \\
&= \left|(\alpha\mathbf{v} + (1-\alpha)\mathbf{v}'')^\top \sigma((\alpha\mathbf{U}(1-\alpha)\mathbf{U}'')\mathbf{x}) - \alpha\mathbf{v}^\top\sigma(\mathbf{U}\mathbf{x}) - (1-\alpha)\mathbf{v}''^\top\sigma(\mathbf{U}''\mathbf{x})\right| \\
&= \left|\alpha\mathbf{v}^\top[\sigma((\alpha\mathbf{U} + (1-\alpha)\mathbf{U}'')\mathbf{x}) - \sigma(\mathbf{U}\mathbf{x})] + (1-\alpha)\mathbf{v}''^\top[\sigma((\alpha\mathbf{U} + (1-\alpha)\mathbf{U}'')\mathbf{x}) - \sigma(\mathbf{U}''\mathbf{x})]\right| \\
&\leq \left|\alpha\mathbf{v}^\top[\sigma((\alpha\mathbf{U} + (1-\alpha)\mathbf{U}'')\mathbf{x}) - \sigma(\mathbf{U}\mathbf{x})]\right| \\
&+ \left|(1-\alpha)\mathbf{v}''^\top[\sigma((\alpha\mathbf{U} + (1-\alpha)\mathbf{U}'')\mathbf{x}) - \sigma(\mathbf{U}''\mathbf{x})]\right|
\end{aligned} \tag{6}$$

Since each element of $\mathbf{v}$ is sampled uniformly from $[-1/\sqrt{h}, 1/\sqrt{h}]$, for any $\mathbf{r} \in \mathbb{R}^h$ we have that $\mathbb{E}[\mathbf{v}^\top\mathbf{r}] = 0$ and by Hoeffding's inequality,

$$P\left(\left|\mathbf{v}^\top\mathbf{r}\right| \geq t\right) \leq 2\exp\left(\frac{-ht^2}{2\|\mathbf{r}\|_2^2}\right) \tag{7}$$

Using the above argument, with probability $1 - \delta/3$, we can bound the right hand side of inequality (6) as

follows:

$$
\begin{aligned}
&\left|\alpha \mathbf{v}^\top[\sigma((\alpha\mathbf{U}+(1-\alpha)\mathbf{U}'')\mathbf{x})-\sigma(\mathbf{U}\mathbf{x})]\right| \\
&+\left|(1-\alpha)\mathbf{v}''^\top[\sigma((\alpha\mathbf{U}+(1-\alpha)\mathbf{U}'')\mathbf{x})-\sigma(\mathbf{U}''\mathbf{x})]\right| \\
&\leq \alpha\sqrt{\frac{2\log(12/\delta)}{h}}\left\|\sigma((\alpha\mathbf{U}+(1-\alpha)\mathbf{U}'')\mathbf{x})-\sigma(\mathbf{U}\mathbf{x})\right\|_2 \qquad (8)\\
&+(1-\alpha)\sqrt{\frac{2\log(12/\delta)}{h}}\left\|\sigma((\alpha\mathbf{U}+(1-\alpha)\mathbf{U}'')\mathbf{x})-\sigma(\mathbf{U}''\mathbf{x})\right\|_2 \\
&\leq \alpha\sqrt{\frac{2\log(12/\delta)}{h}}\left\|(\alpha\mathbf{U}+(1-\alpha)\mathbf{U}'')\mathbf{x}-\mathbf{U}\mathbf{x}\right\|_2 \qquad (9)\\
&+(1-\alpha)\sqrt{\frac{2\log(12/\delta)}{h}}\left\|(\alpha\mathbf{U}+(1-\alpha)\mathbf{U}'')\mathbf{x}-\mathbf{U}''\mathbf{x}\right\|_2 \\
&=\alpha\sqrt{\frac{2\log(12/\delta)}{h}}\left\|(1-\alpha)(\mathbf{U}-\mathbf{U}'')\mathbf{x}\right\|_2+(1-\alpha)\sqrt{\frac{2\log(12/\delta)}{h}}\left\|\alpha(\mathbf{U}-\mathbf{U}'')\mathbf{x}\right\|_2 \\
&\leq \sqrt{\frac{\log(12/\delta)}{2h}}\left\|(\mathbf{U}-\mathbf{U}'')\mathbf{x}\right\|_2 \qquad (10)
\end{aligned}
$$

where the inequality 9 is due to Lipschitz property of ReLU activations. Now, all we need to do is to bound $\|(\mathbf{U}-\mathbf{U}'')\mathbf{x}\|_2$. Note that for any $(i,j)\in[h]\times[d]$, $u_{ij}-u''_{ij}$ is an independent random variable with mean zero and bounded magnitude($2/\sqrt{d}$ if $i\in I$ and 2ξ otherwise). Therefore, we can again use the Hoffding's inequality similar to inequality (7) for each row i and after taking a union bound, we have the following inequality with probability $1-\delta/3$,

$$
\begin{aligned}
\left\|(\mathbf{U}-\mathbf{U}'')\mathbf{x}\right\|_2 &= \sqrt{\sum_{i\in I}(\mathbf{u}_i-\mathbf{u}''_i)\mathbf{x}+\sum_{i\in[h]\setminus I}(\mathbf{u}_i-\mathbf{u}''_i)\mathbf{x}} \\
&\leq \|x\|_2\sqrt{|I|\frac{4\log(12h/\delta)}{d}+(h-|I|)(4\xi^2\log(12h/\delta)} \\
&\leq 2\sqrt{\log(12h/\delta)\left(|I|+\xi^2dh\right)}
\end{aligned}
$$

Where the last inequality is using $\|\mathbf{x}\|_2=\sqrt{d}$. Substituting the above inequality into the right hand side of the inequality (10), gives us the following upper bound on the left hand side of the inequality in the theorem statement:

$$
\begin{aligned}
&\left|f_{\alpha\mathbf{v}+(1-\alpha)\mathbf{v}'',\alpha\mathbf{U}+(1-\alpha)\mathbf{U}''}(\mathbf{x})-\alpha f_{\mathbf{v},\mathbf{U}}(\mathbf{x})-(1-\alpha)f_{\mathbf{v}',\mathbf{U}'}(\mathbf{x})\right| \\
&\leq \sqrt{\frac{\log(12/\delta)}{2h}}\left\|(\mathbf{U}-\mathbf{U}'')\mathbf{x}\right\|_2 \qquad (11)\\
&\leq \sqrt{2\log(12/\delta)\log(12h/\delta)\left(\frac{|I|}{h}+\xi^2d\right)} \qquad (12)
\end{aligned}
$$

Setting $\xi=\epsilon/\sqrt{4d\log(12/\delta)\log(12h/\delta)}$, gives the following bound on h:

$$
\begin{aligned}
h &\leq \frac{4\log(12/\delta)\log(12h/\delta)|I|}{\epsilon^2} \\
&\leq \frac{4\log(12/\delta)\log(12h/\delta)|S_\xi|\sqrt{\frac{h}{2}\log(12|S_\xi|/\delta)}}{\epsilon^2}
\end{aligned}
$$

Therefore, we have:

$$h \leq \left(\frac{4 \log(12/\delta) \log(12h/\delta) |S_\xi| \sqrt{\log(12|S_\xi|/\delta)}}{\epsilon^2} \right)^2$$
$$\leq \left(\frac{4 \log(12/\delta) \log(12h/\delta)}{\epsilon^2} \right)^{d+2} (\log(12/\delta) + d \log(1/\epsilon))$$

Using the above inequality, we have $\epsilon = \tilde{O}(h^{-\frac{1}{2d+4}})$. □

C Additional plots

C.1 Small networks

We consider MLPs with the same architecture starting from 100 different initializations and trained on the MNIST dataset. For each pair of the networks we calculate their loss barrier and plot the histogram on the values. Next for each pair of the networks, we find the permutation that minimizes the barrier between them and plot the histogram for all the pairs. We do this investigation for different network sizes. The results are shown in Figure 11 top row. Note that since we consider all possible pairs, the observed barrier values are not i.i.d. If instead we randomly divide the 100 trained networks into two sets and choose pairs that are made by picking one network from each set, we will have an i.i.d sampling strategy. We investigate the pairs of networks trained on MNIST and measure the value of the direct and indirect barriers between them. *Indirect barrier* between two networks A, C is minimum over all possible intermediate points B of maximum of barrier between A, B and barrier between B, C, *i.e.*,

$$\min_{B!=A, B!=C} \max(B(\theta_A, \theta_B), B(\theta_B, \theta_C)).$$

The reason we look into this value is that if maximum between two barriers is small, it means that both barriers are small and therefore there exist an indirect path between A and C.

C.2 Loss barriers on the test set

Width. We evaluate the impact of width on the test barrier size in Figure 12. In comparison to Figure 2 the magnitude of test barriers are shifted to lower values as the test accuracy is lower than train accuracy. This effect is intensified for harder tasks such as CIFAR100. The double descent phenomena is also observed here, especially for simpler tasks, *e.g.*, MNIST and SVHN.
Depth. We evaluate the impact of depth on the test barrier size in Figure 13. For MLPs, we fixed the layer width at 2^{10} while adding identical layers as shown along the x-axis. Similar to Figure 3 we observe a fast and significant barrier increase as more layers are added. In comparison to Figure 3 the magnitude of test barriers are shifted to lower values as the test accuracy is lower than train accuracy.

C.3 Similarity of $\mathcal{S}$ and $\mathcal{S}'$

We note SA success on test barrier removal by comparing Figure 14 and Figure 15. SA performance on $\mathcal{S}$ and $\mathcal{S}'$ yields similar results for both before and after permutation scenarios. Such similar performance is observed along a wide range of width and depth for both MLP and Shallow-CNN over different datasets(MNIST, SVHN, CIFAR10, CIFAR100). We look into effects of changing model size in terms of width and depth as in earlier Sections, and note that similar trends hold for before and after permuting solution θ_1.

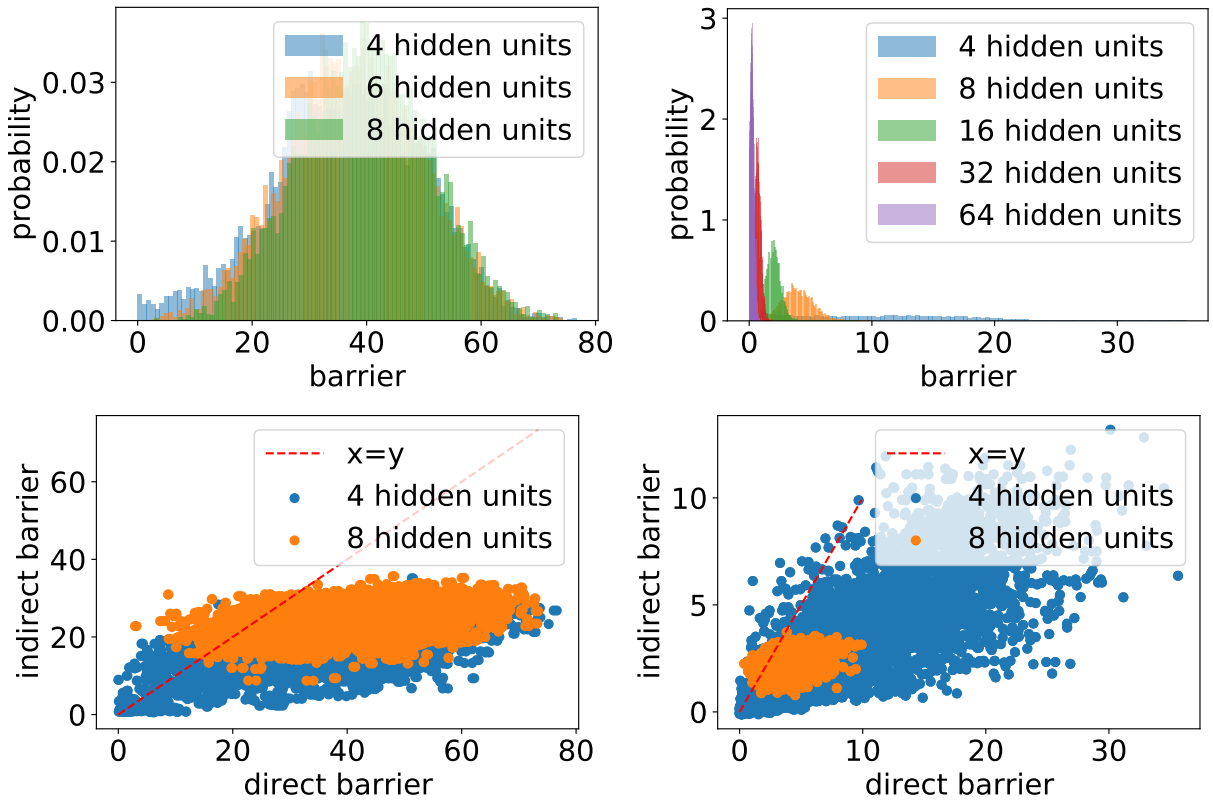


Figure 11: **Histogram of barrier values between pairs of 100 networks with the same architecture trained on MNIST starting from different initializations.** We find a permutation for each pair that minimizes the barrier. This is done for two layer MLPs and repeated for networks of different sizes. Bottom row: Indirect barrier between two networks A,C is minmimum over all possible intermediate points B of maximum of barrier between A, B and barrier between B, C. **Left:** before the permutation; **Right:** after the permutation.

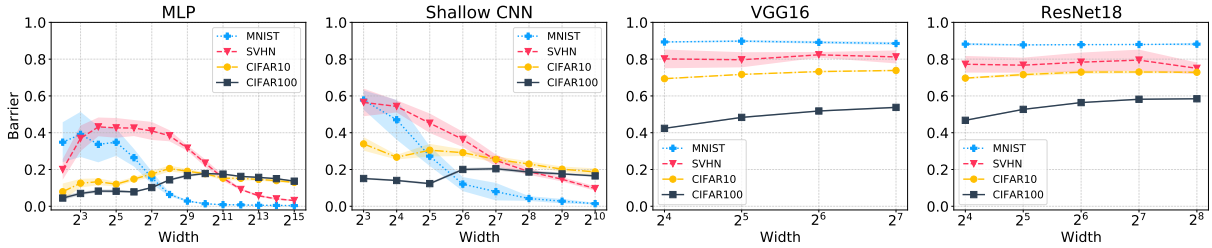


Figure 12: **Effect of width on barrier size (Test).** From **left to right**: one-layer MLP, two-layer Shallow-CNN, VGG-16, and ResNet-18 architectures and MNIST, CIFAR-10, SVHN, CIFAR-100 datasets. When the task is hard (CIFAR10, CIFAR100) the test barrier shrinks. For simpler tasks and large width sizes also the barrier becomes small.

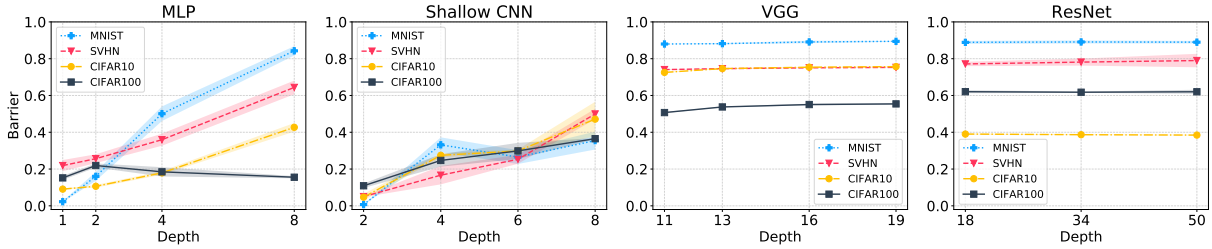


Figure 13: **Effect of depth on barrier size (Test).** From **left to right** one-layer MLP, two-layer Shallow-CNN, VGG(11,13,16,18), and ResNet(18,34,50) architectures and MNIST, CIFAR-10, SVHN, CIFAR-100 datasets. For MLP and Shallow-CNN, we fixed the layer width at 2^{10} while adding identical layers as shown along the x-axis. Similar behavior is observed for fully connected and CNN family, *i.e.*, low barrier when number of layers are low while we observe a fast and significant barrier increase as more layers are added. Increasing depth leads to higher barrier values until it saturates (as seen for ResNet).

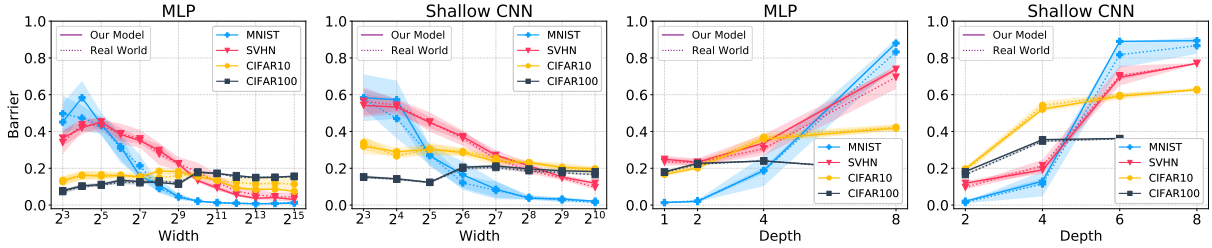


Figure 14: **Effect of width and depth on barrier consistency between real world and our model *before* permutation (Test)**. Search space is reduced here *i.e.*, we take two SGD solutions θ_1 and θ_2, permute θ_1 and report the barrier between permuted θ_1 and θ_2 as found by SA with $n = 2$. Similarity of loss barrier between real world and our model is preserved across model type and dataset choices as width and depth of the models are increased.

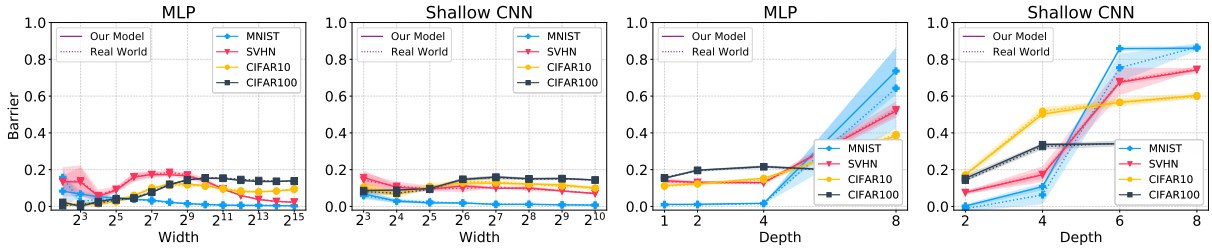


Figure 15: **Effect of width and depth on barrier consistency between real world and our model *after* permutation (Test).** We observe that reducing the search space makes SA more successful in finding the permutation $\{\pi\}$ to remove the barriers. Specifically, SA could indeed find permutations across different networks and datasets that when applied to θ_1 result in almost zero test barrier *e.g.*, MLP across MNIST dataset where depth is 1 and width is larger than 2^6, MLP for MNIST where depth is 2 and 4, and width is 2^{10}, Shallow-CNN for MNIST where depth is 2, Shallow-CNN for SVHN where depth is 2 and width is 2^{10}.