Git Re-Basin: Merging Models modulo Permutation Symmetries

Motivation

- Why are SGD so effective in deep learning?
- When linearly interpolation between initialization and final trained weights, why does loss smoothly and monotonically decrease?
- How can two independently trained models achieve nearly identical performance? Especially, why do their loss curves look identical?

Motivation

- Why are SGD so effective in deep learning?
- When linearly interpolation between initialization and final trained weights, why does loss smoothly and monotonically decrease?
- **How can two independently trained models achieve nearly identical performance? Especially, why do their loss curves look identical?**

Invariance of Training Dynamics

Conjecture (Permutation invariance, informal)

Most SGD solutions belong to a set whose elements can be permuted so that zero barrier exists on the linear interpolation between any two permuted elements. Refer to such solutions as being *linearly mode connected* (LMC).

Definition (Loss barrier)

Given two points Θ_A, Θ_B such that $\mathcal{L}(\Theta_A) \simeq \mathcal{L}(\Theta_B)$, the *loss barrier* is defined as

$$\max_{\lambda \in [0,1]} \mathcal{L}((1-\lambda)\Theta_A + \lambda\Theta_B) - \frac{1}{2}\left(\mathcal{L}(\Theta_A) + \mathcal{L}(\Theta_B)\right).$$

Note here loss barrier is non-negative, and zero indicates $\mathcal{L}(\Theta_A)$ and $\mathcal{L}(\Theta_B)$ can be interpolated by a flat line or positive curvature curve.

Mode Connectivity

Local optimas for deep neural networks are connected by very simple curves, such as polygon chain.

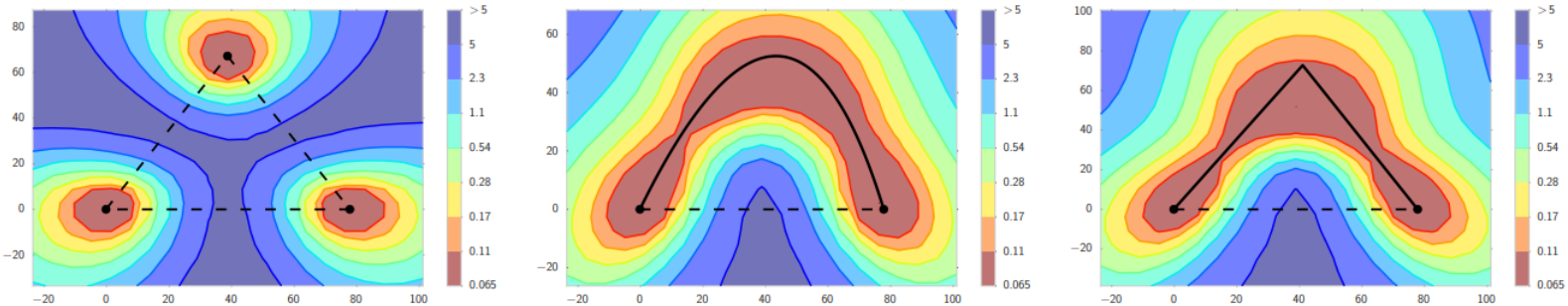

Figure 1: segment (left), Bezier curve (middle), Polygon chain (right)

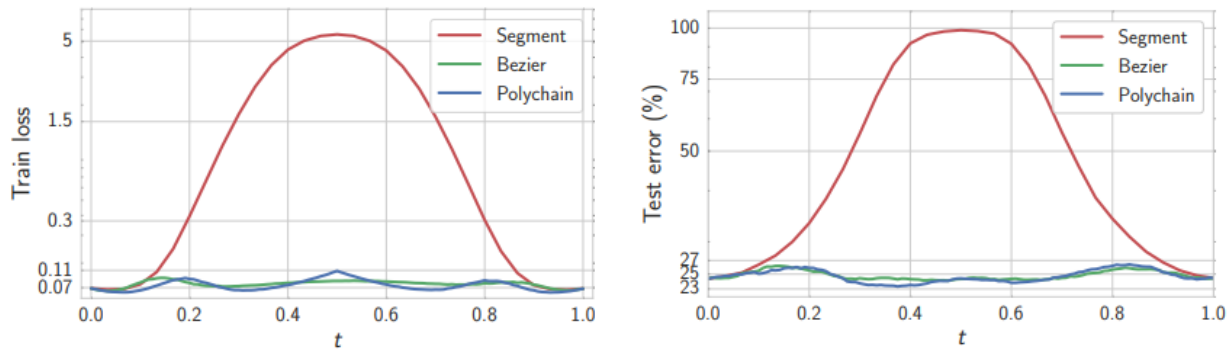

Figure 2: Train loss (left) and Test error (right) along specified curves

Setup

For simplicity, consider L-layered MLP,

$$f(x, \Theta) = z_{L+1}, \quad z_{\ell+1} = \sigma(W_\ell z_\ell + b_\ell), \quad z_1 = x,$$

where x is an input, $\Theta = (W_1, b_1, \cdots W_L, b_L)$, and σ denotes an element-wise nonlinear activation function. Also, consider a loss, $\mathcal{L}(\Theta)$, that measures the performance of the weight Θ.

Permutation Symmetries of Weight Space

Consider a permutation matrix $\mathbf{P} \in S_d$, where S_d is the set of all $d \times d$ permutation matrices, or the symmetric group. Then with any intermediate output $z_{\ell+1}$ of ℓth layer, we have

$$z_{\ell+1} = \mathbf{P}^\intercal \mathbf{P} z_{\ell+1} = \mathbf{P}^\intercal \mathbf{P} \sigma\left(W_\ell z_\ell + b_\ell\right) = \mathbf{P}^\intercal \sigma\left(\mathbf{P} W_\ell z_\ell + \mathbf{P} b_\ell\right).$$

Using above equalities, define Θ' from Θ by keeping all parameters same, except

$$W'_\ell = \mathbf{P} W_\ell, \quad b'_\ell = \mathbf{P} b_\ell, \quad W'_{\ell+1} = W_{\ell+1} \mathbf{P}^\intercal.$$

Then two models parameterized by Θ and Θ', would be functionally equivalent, i.e. $f\left(x, \Theta\right) = f\left(x, \Theta'\right)$.
Denote $\pi(\Theta)$ as such *functionality-preserving permutation* of Θ.

Rewriting the Conjecture

Let Θ_A, Θ_B be the weights of two independently trained models, A and B respectively. Then there exists a functionality-preserving permutation $\pi(\Theta_B)$ such that Θ_A and $\pi(\Theta_B)$ are LMC.

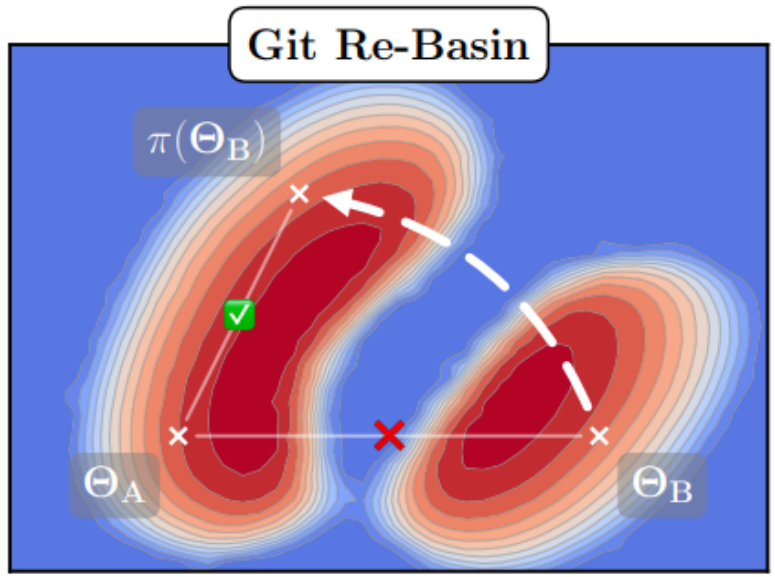

Figure 3: Visualization of the conjecture

Permutation Selection Methods

However, naively exploring all permutation symmetries is impossible.

Architecture	**Num. Permutation Symmetries**
MLP (3 layers, 512 width)	$10 \wedge 3498$
VGG16	$10 \wedge 35160$
ResNet50	$10 \wedge 55109$

Table 1: Numbers of permutation symmetries of DNNs are very large.

Moreover, evaluating the loss barrier could also be very expensive.

Permutation Selection Methods

Three methods are proposed to cope with these problems:

- Matching Activations
- Matching Weights
- Learning Permutations with a Straight-Through Estimator

Matching Activations

Assumption

Neural network units that fire together, wire together.

Method

Match the activations of model A and B with the regression framework by constraining ordinary least squares. For the intermediate outputs of ℓth layer, $Z^{(A)}, Z^{(B)} \in \mathbf{R}^{d \times n}$, choose $\mathbf{P}_\ell$ such that

$$\mathbf{P}_\ell = \arg\min_{\mathbf{P} \in S^d} \sum_{i=1}^{n} \left\| Z_{:,i}^{(A)} - \mathbf{P} Z_{:,i}^{(B)} \right\|^2 \tag{1}$$

$$= \arg\max_{P \in S^d} \sum_{i=1}^{n} \left\langle Z_{:,i}^{(A)}, \mathbf{P} Z_{:,i}^{(B)} \right\rangle = \arg\max_{\mathbf{P} \in S^d} \left\langle \mathbf{P}, Z^{(A)} \left(Z^{(B)} \right)^{\mathsf{T}} \right\rangle_F,$$

where $\langle A, B \rangle_F$ is the Frobenius norm.

Matching Activations

The equation (1) is a *linear assignment problem* (LAP), for which efficient and practical algorithms are known: Hungarian algorithm $O(n^4)$, Jonker-Volgenant algorithm $O(n^3)$.

Definition

Linear assignment problem is a fundamental combinatorial optimization problem. In most general form, the problem consists of a number of agents and a number of jobs. To solve the problem one need to find a 1-1 correspondence between agents and jobs that minimize the assignment cost. For example, in clustering problem, assigning class label to each cluster can be viewed as LAP.

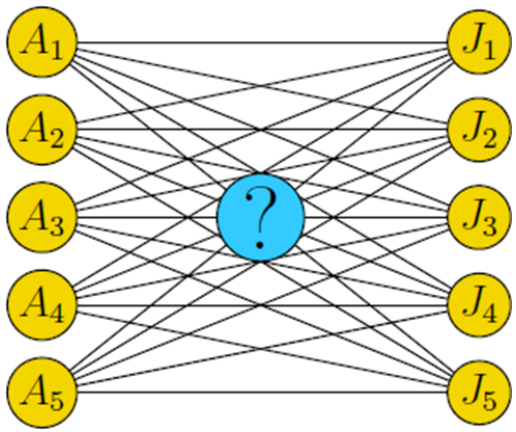

Figure 4: Typical LAP

Matching Weights

Assumption

If $\left[W_\ell^{(A)}\right]_i$ and $\left[W_\ell^{(B)}\right]_j$ were equal, then they would compute exactly the same features, and hence would be associated.

Method (Naive)

$$\begin{aligned}\underset{\pi=\{\mathbf{P}_i\}}{\arg\max} \left\langle W_1^{(A)}, \mathbf{P}_1 W_1^{(B)} \right\rangle_F + \left\langle W_2^{(A)}, \mathbf{P}_2 W_2^{(B)} \mathbf{P}_1^\intercal \right\rangle_F \\ + \cdots + \left\langle W_L^{(A)}, W_L^{(B)} \mathbf{P}_{L-1}^\intercal \right\rangle_F\end{aligned} \tag{2}$$

However equation (2), coined as *sum of bilinear assignments problem* (SOBLAP), is NP-hard. Contrast to previous LAP, permutations of both rows AND columns should be considered, making it difficult to solve.

Matching Weights

However, if we focus on a single $\mathbf{P}_\ell$, the problem can be reduced to a LAP,

$$\begin{aligned}
&\arg\max_{\mathbf{P}_\ell} \left\langle W_\ell^{(A)}, \mathbf{P}_\ell W_\ell^{(B)} \mathbf{P}_{\ell-1}^\intercal \right\rangle_F + \left\langle W_{\ell+1}^{(A)}, \mathbf{P}_{\ell+1} W_{\ell+1}^{(B)} \mathbf{P}_\ell{}^\intercal \right\rangle_F \\
&= \arg\max_{\mathbf{P}_\ell} \left\langle \mathbf{P}_\ell, W_\ell^{(A)} \mathbf{P}_{\ell-1} \left(W_\ell^{(B)}\right)^\intercal + \left(W_{\ell+1}^{(A)}\right)^\intercal \mathbf{P}_{\ell+1} W_{\ell+1}^{(B)} \right\rangle_F
\end{aligned}$$

Matching Weights

Method (Permutation Coordinate Descent)

Algorithm 1: Permutation Coordinate Descent

Initialize: $\mathbf{P}_1 \leftarrow I, \cdots \mathbf{P}_{L-1} \leftarrow I$

while *not converge* **do**

  for $\ell \in \texttt{RandomPermutation}(1, \cdots, L-1)$ **do**

    $$P_\ell \leftarrow \texttt{SolveLAP}\left(W_\ell^{(A)} \mathbf{P}_{\ell-1} \left(W_\ell^{(B)}\right)^\intercal + \left(W_{\ell+1}^{(A)}\right)^\intercal \mathbf{P}_{\ell+1} W_{\ell+1}^{(B)}\right);$$

  end

end

Note here weight matching ignores the data distribution entirely. However surprisingly, it shows competitive results compared to the data-aware methods.

Matching Weights

Lemma 1

Algorithm 1 terminates.

Proof.

Consider a DAG whose vertex is each possible permutation $\pi_i = \{\mathbf{P}_1, \cdots \mathbf{P}_{L-1}\}$, and edge $\pi_i \to \pi_j$ if π_j can be reached from π_i by updating single permutation $\mathbf{P}_\ell$ (as in Algorithm 1). Let $\rho(\pi) = \text{vec}(\Theta_A) \cdot \text{vec}(\pi(\Theta_B))$. Then $\pi_i \to \pi_j$ implies that $\rho(\pi_i) < \rho(\pi_j)$. Now suppose that Algorithm 1 does not converge. Since there exist only finitely many vertices, this implies that there exists a cycle. Contradiction. $\square$

Learning Permutations with a STE

Method

"Learn" $\tilde{\Theta}_B$ such that

$$\min_{\tilde{\Theta}_B} \mathcal{L}\left(\frac{1}{2}\left(\Theta_A + \text{proj}\left(\tilde{\Theta}_B\right)\right)\right),$$

where

$$\text{proj}(\Theta) = \arg\max_{\pi} \text{vec}(\Theta) \cdot \text{vec}(\pi(\Theta_B)).$$

However, the projection operator $\text{proj}(\cdot)$ is non-differentiable.

Straight-Through Estimator (STE)

Consider a neural network with a binarize activation f, for example,

$$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0. \end{cases}$$

Then the gradient of such activation is 0. One popular solution for this is to use Straight-Through Estimator that estimates the gradient of function by bypassing the activation function.

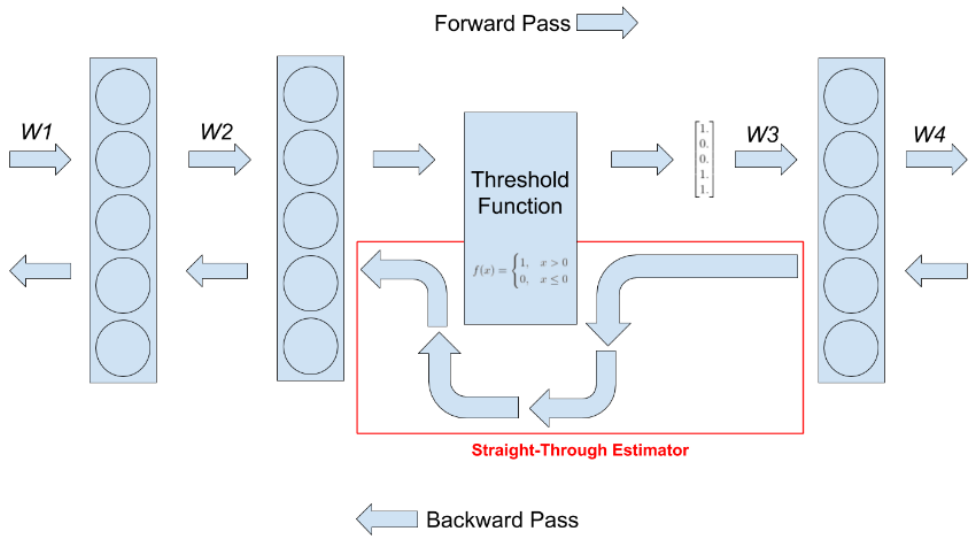

Figure 5: Visualization of STE

Straight-Through Estimator (STE)

Similarly, when learning the permutations, we can use the bypass, or STE. In the forward pass, we use $\text{proj}\left(\tilde{\Theta}_B\right)$ to compute the loss, and in the backward pass, we bypass the projection and use $\tilde{\Theta}_B$ to compute the gradient.

Learning Permutations with a STE

Although "learning" the permutation utilize the advantages of both activation matchings and weight matchings, it comes at a very steep computational cost.

Experiment

In all cases, permuting the weights significantly improved over the naive interpolation. As expected the STE matching performed the best (by surprisingly small margin), however at the great cost. Activation matching and weight matching performed similarly, but the latter is faster and does not require the data.

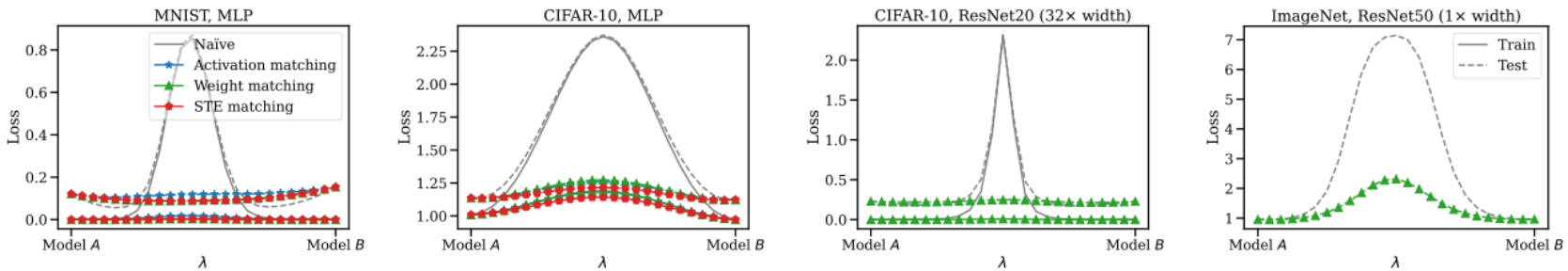

Figure 6: Loss when interpolating between two models trained on MNIST, CIFAR10, and ImageNet

Experiment

Although one could guess that entire weight space is in a single basin modulo permutation symmetry, it is not true. The LMC seems to be a property of SGD training.

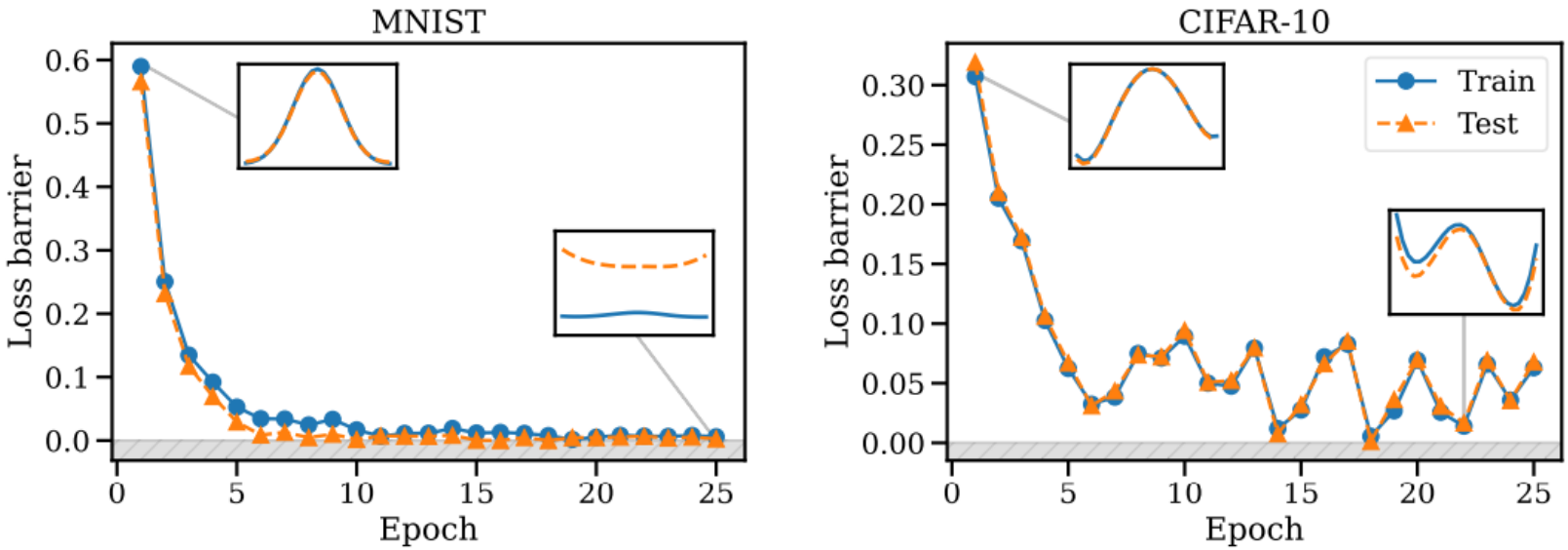

Figure 7: Loss barrier dynamics as the training proceeds.

Experiment

It is a conventional wisdom that wider architecture is easier to optimze. Similarly there is a clear relationship between model width and linear mode connectivity.

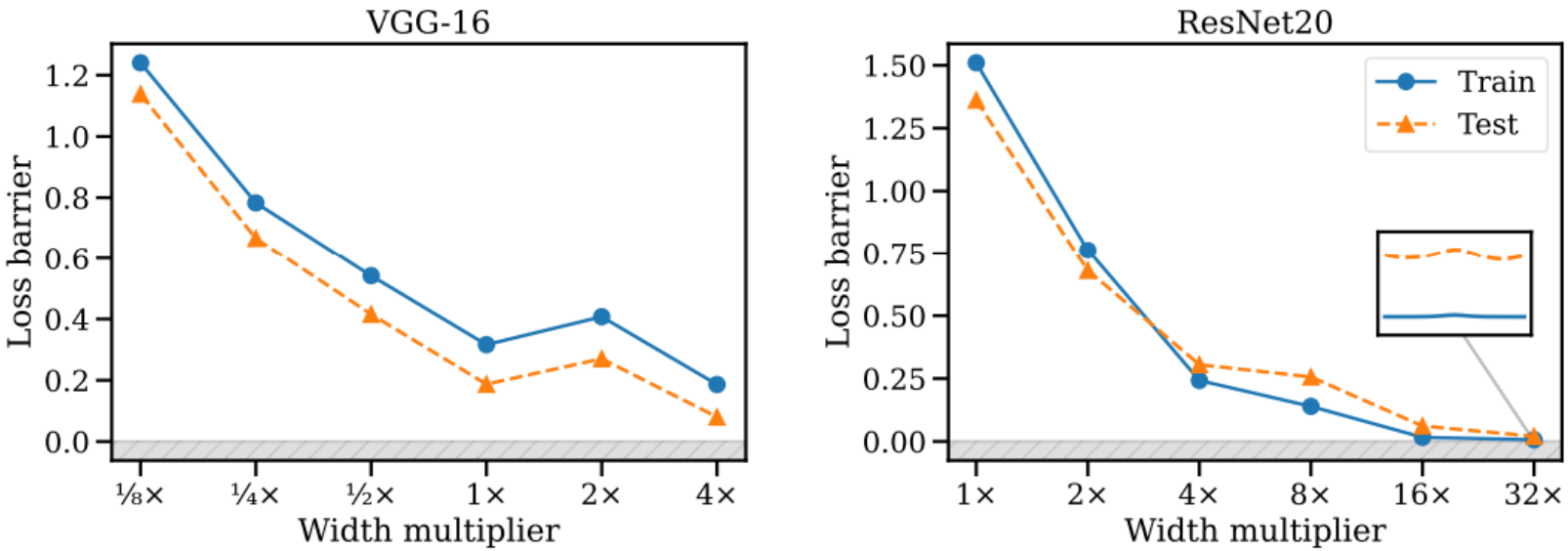

Figure 8: Loss barrier and width of the model

Experiment

Consider the case where models can be trained on each dataset separately, but cannot train in aggregate. Can we merge these models so that it can perform well on the whole dataset?

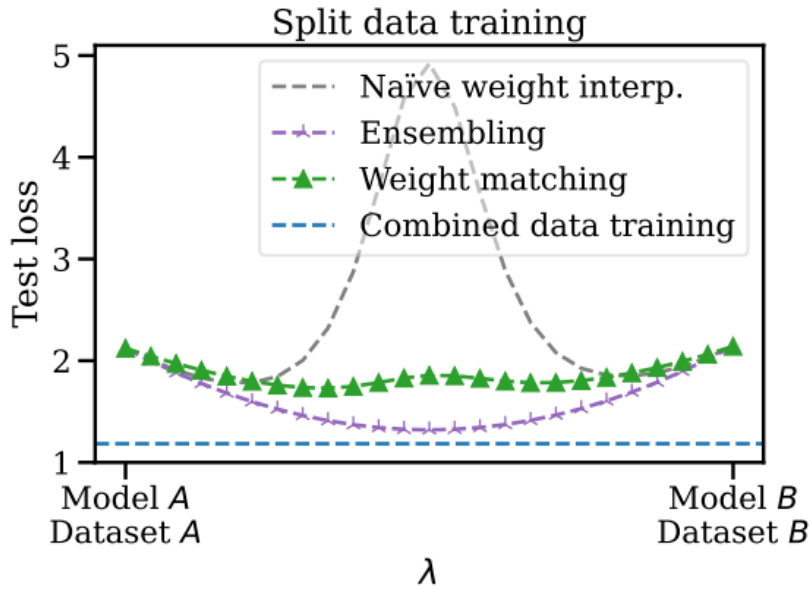

Figure 9: Merging models trained on disjoint datasets

Model A is trained on dataset A, containing 20% of samples labelled 0-49, 80% of samples labelled 50-99 of CIFAR-100, and Model B is trained on dataset B, that contains rest of the dataset.

Counterexample

Consider a 2-dimensional binary classification task:

$$x \sim \text{Unif}\left([-1,1]^2\right), \quad y = \mathbf{1}_{x_1<0 \text{ and } x_2>0}$$

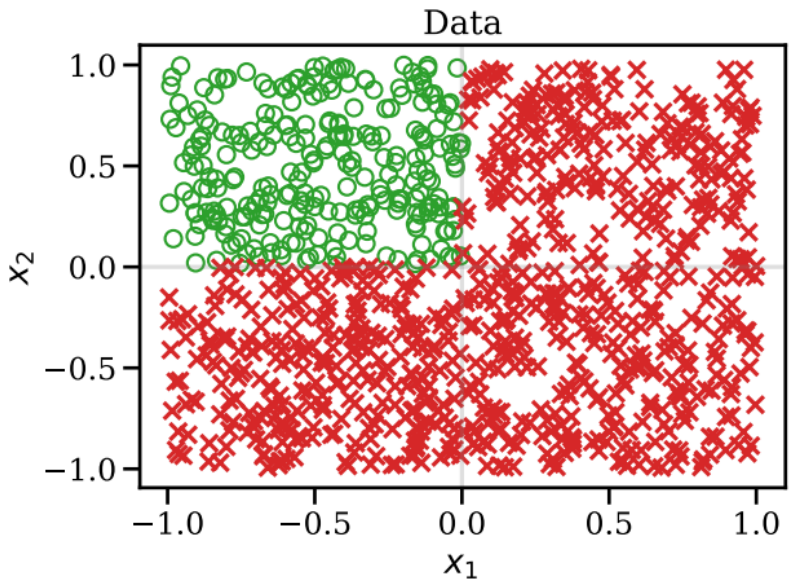

Figure 10: Dataset visualization

Counterexample

Following 2-layered MLPs with 2 hidden units with ReLU activation can perfectly fit the data:

$$f_A(x) = \begin{bmatrix} -1 & -1 \end{bmatrix} \sigma\left(\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \sigma\left(\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$$

$$f_B(x) = \begin{bmatrix} -1 & -1 \end{bmatrix} \sigma\left(\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \sigma\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$$

However there exists no permutation that satisfies LMC.

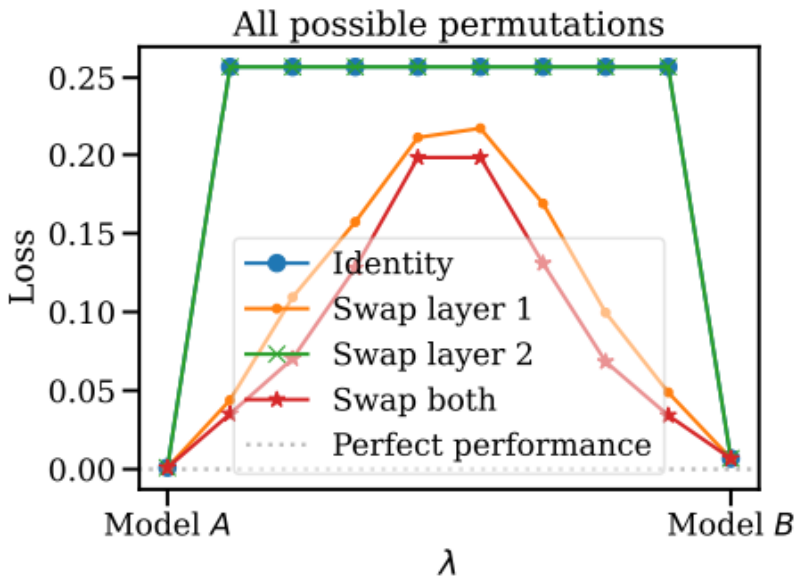

Figure 11: Counterexample

Evaluation

ICLR 2023 Review: 8/10, 8/10, 10/10

"Dear authors, $\cdots$ *I believe that the main contribution of this paper is not algorithmic, but rather conceptual* $\cdots$ *I believe that this paper provides an interesting new perspective on the structure of the loss surfaces of neural networks* $\cdots$ I still believe that the paper brings value to the community and should be accepted to the conference."