Embedding Principle of Loss Landscape of Deep Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Understanding the structure of loss landscape of deep neural networks (DNNs) is obviously important. In this work, we prove an embedding principle that the loss landscape of a DNN "contains" all the critical points of all the narrower DNNs. More precisely, we propose a critical embedding such that any critical point, e.g., local or global minima, of a narrower DNN can be embedded to a critical point/hyperplane of the target DNN with higher degeneracy and preserving the DNN output function. The embedding structure of critical points is independent of loss function and training data, showing a stark difference from other nonconvex problems such as protein-folding. Empirically, we find that a wide DNN is often attracted by highly-degenerate critical points that are embedded from narrow DNNs. The embedding principle provides an explanation for the general easy optimization of wide DNNs and unravels a potential implicit low-complexity regularization during the training. Overall, our work provides a skeleton for the study of loss landscape of DNNs and its implication, by which a more exact and comprehensive understanding can be anticipated in the near future.

1 Introduction

Understanding the loss landscape of DNNs is essential for a theory of deep learning. An important problem is to quantify exactly how the loss landscape looks like (E et al., 2020). This problem is difficult since the loss landscape is so complicated that it can almost be any pattern (Skorokhodov and Burtsev, 2019). Moreover, its high dimensionality and the dependence on data, model and loss make it very difficult to obtain a general understanding through empirical study. Therefore, though it has been extensively studied over the years, it remains an open problem to provide a clear picture about the organization of its critical points and their properties.

In this work, we make a step towards this goal through proposing a very general embedding operation of network parameters from narrow to wide DNNs, by which we prove an embedding principle for fully-connected DNNs stated *intuitively* as follows:

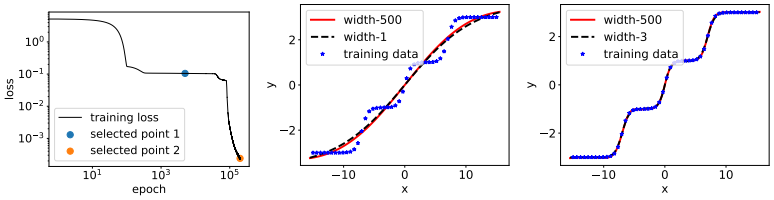
Embedding principle*: the loss landscape of any network "contains" all critical points of all narrower networks.*

A "narrower network" means a DNN of the same depth but width of each layer no larger than the target DNN. The embedding principle slightly abuses the notion of "contain" since parameter space of DNNs of different widths are different. However, this inclusion relation is reasonable in the sense that, by our embedding operation, any critical point of any narrower network can be embedded to a critical point of the target network preserving its output function. Because of this criticality preserving property, we call this embedding operation the critical embedding.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We conclude our study by a "principle" since the embedding principle is a very general property of loss landscape of DNNs independent of the training data and choice of loss function, and is intrinsic to the layer-wise architecture of DNNs. In addition, the embedding principle is closely related to the training of DNNs. For example, as shown in Fig. 1(a), the training of a width-500 two-layer tanh NN experiences stagnation around the blue dot presumably very close to a saddle point, where the loss decreases extremely slowly. As shown in Fig. 1(b), we find that the DNN output at this blue point (red solid) is very close to the output of the global minimum (black dashed) of the width-1 NN, indicating that the underlying two critical points of two DNNs with different widths have the same output function conforming with the embedding principle. Importantly, this example shows that the training of a wide DNN can indeed experience those critical points from a narrow DNN unraveled by the embedding principle. Moreover, it demonstrates the potential of a transition from a local/global minimum of a narrow NN to a saddle point of a wide NN, which may be the reason underlying the easy optimization of wide NNs.

The embedding principle suggests an underlying mechanism to understand why heavily overparameterized DNNs often generalize well (Breiman, 1995; Zhang et al., 2017) as follows. Roughly, the overparameterized DNN has a large capacity, which seems contradictory to the conventional learning theory, i.e., learning by a model of large capacity easily leads to overfitting. The embedding principle shows that the optima of a wide network intrinsically may be embedded from an optima of a much narrower network, thus, its effective capacity is much smaller. For example, as illustrated in Fig. 1, training of a heavily overparametrized width-500 NN (vs. 50 training data) with small initialization first stagnated around a saddle presumably from width-1 NN and later converges to a global minimum presumably from width-3 NN, which clearly does not overfit. This implicit regularization effect unraveled by the embedding principle is consistent with previous works, such as low-complexity bias (Arpit et al., 2017; Kalimeris et al., 2019; Jin et al., 2020), low-frequency bias (Xu et al., 2019, 2020; Rahaman et al., 2019), and condensation phenomenon of network weights (Luo et al., 2021; Chizat and Bach, 2018; Ma et al., 2020).



(a) (b) (c)

Figure 1: (a) The training loss of two-layer tanh neural network with 500 hidden neurons. (b) (c) Red solid: the DNN output at a training step indicated by (b) the blue dot or (c) the orange dot in (a); Black dashed: the output of the global minimum of (b) width-1 DNN or (c) width-3 DNN, respectively; Blue dots: training data.

2 Related works

The loss landscape of DNNs is complex and related to the generalization. Skorokhodov and Burtsev (2019) numerically show that the loss landscape can almost be any pattern. Keskar et al. (2017) visualize minimizers in a 1d slice and suggest that a flat minimizer generalizes better. Wu et al. (2017) find that the volume of basin of attraction of good minima may dominate over that of poor minima in practical problems. He et al. (2019) show that at a local minimum there exist many asymmetric directions such that the loss increases abruptly along one side, and slowly along the opposite side.

Degeneracy is also an important property of minima. Cooper (2021) shows that global minima is typically a high dimensional manifold for overparameterized DNNs. Sagun et al. (2016) empirically shows that Hessian of the minimizer obtained by the training has many zero eigenvalues. Under strong assumptions, Choromanska et al. (2015) shows minima tend to be highly degenerate. This work demonstrates wide existence of highly degenerate critical points, including local or global minima and saddle points, in the loss landscape by the embedding principle.

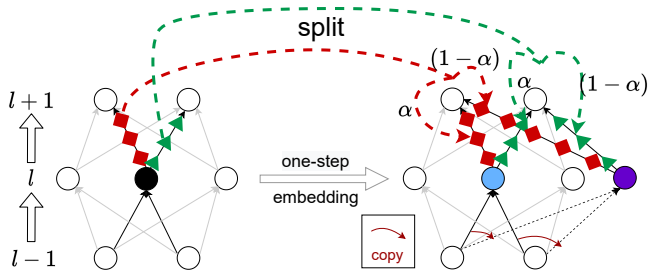


Figure 2: Illustration of one-step embedding. The black neuron in the left network is splitted into the blue and purple neurons in the right network. The red (green) output weight of the black neuron in the left net is splitted into two red (green) weights in the right net with ratio α and $(1-\alpha)$, respectively.

Lots of previous theoretical works focus on very wide DNNs, such as the phase diagram of two-layer ReLU infinite-width NNs (Luo et al., 2021), NTK regime (Jacot et al., 2018; Arora et al., 2019; Zhang et al., 2020; Du et al., 2019; Zou et al., 2018; Allen-Zhu et al., 2019; E et al., 2019), mean-field regime (Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018; Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2020). By the embedding principle, this work demonstrate the loss landscape similarity between a moderate-width NN and a very wide NN, that they share a set of critical points embedded from that of narrower NNs. Therefore, results about infinite-width NNs could provide valuable insights about training of finite-width NNs used in practice.

The complexity of NN output increases during the training (Arpit et al., 2017; Xu et al., 2019, 2020; Rahaman et al., 2019; Kalimeris et al., 2019; Goldt et al., 2020; He et al., 2020; Mingard et al., 2019; Jin et al., 2020). For example, the frequency principle (Xu et al., 2019, 2020) states that DNNs often fit target functions from low to high frequencies during the training.

3 Main results

In this section, we intuitively summarize our key theoretical results about the embedding principle and empirically demonstrate its relevance to practice, starting from proposing an embedding operation as follows. Rigorous theoretical description and proofs are presented in the latter sections.

3.1 Characteristics of embedding principle

Consider a neural network $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{\theta}$ is the set of all network parameters, $\boldsymbol{x} \in \mathbb{R}^d$ is the input. The loss for the training data $S = \{(\boldsymbol{x}_i, \boldsymbol{f}^*(\boldsymbol{x}_i)\}_{i=1}^n$ is denoted by $R_S(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{f}^*(\boldsymbol{x}_i))$.

Definition 1 (critical point). *Parameter vector $\boldsymbol{\theta}$ is a critical point of the landscape of R_S if $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}) = \mathbf{0}$.*

Definition 2 (degree of degeneracy). *The degree of degeneracy of point $\boldsymbol{\theta}$ in the landscape of R_S is the corank of $\nabla_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}} R_S$, i.e., the multiplicity of the zero eigenvalues.*

Note that, above definition of degree of degeneracy is convenient, but requires twice differentiable activation and loss. This condition is actually not necessary for all our theoretical results below. Then, we introduce one-step embedding intuitively, and leave the rigorous definition latter. As shown in Fig. 2, an one-step embedding is performed by splitting any hidden neuron, say the black neuron in the left network, into two neurons colored in blue and purple in the right network. The input weights of the two splitted neurons are the same as the input weights of the original black neuron. Each output weight of the original black neuron is splitted into two parts of fraction α and $(1-\alpha)$ ($\alpha \in \mathbb{R}$, a hyperparameter), respectively. The multi-step embedding is the composition of multiple one-step embeddings. Since each one-step embedding can add one neuron to a selected layer, parameter of any NN can be embedded to the parameter space of any wider NN through a multi-step embedding. The multi-step embedding operation leads to the following property readily.

Proposition (one-step embedding preserves network properties, informal Prop. 1). *For any point $\boldsymbol{\theta}_{\text{narr}}$ of a DNN, a point $\boldsymbol{\theta}_{\text{wide}}$ of a wider DNN obtained from $\boldsymbol{\theta}_{\text{narr}}$ by one-step embedding*

satisfies
(i) $\boldsymbol{f}_{\boldsymbol{\theta}_{\text{narr}}}(\boldsymbol{x}) = \boldsymbol{f}_{\boldsymbol{\theta}_{\text{wide}}}(\boldsymbol{x})$ *for any* $\boldsymbol{x}$*;*
(ii) representation of the wide DNN at $\boldsymbol{\theta}_{\text{wide}}$*, i.e., the set of all different response functions of neurons, is the same as representation of the narrow DNN at* $\boldsymbol{\theta}_{\text{narr}}$*.*

The most important property of this embedding is criticality preserving as follows.

Theorem (criticality preserving, informal Theorem 1). *For any critical point* $\boldsymbol{\theta}_{\text{narr}}$ *of a DNN, a point* $\boldsymbol{\theta}_{\text{wide}}$ *of a wider DNN obtained from* $\boldsymbol{\theta}_{\text{narr}}$ *by multi-step embedding is a critical point.*

The embedding operation explains the cause of a type of degeneracy in the loss landscape.

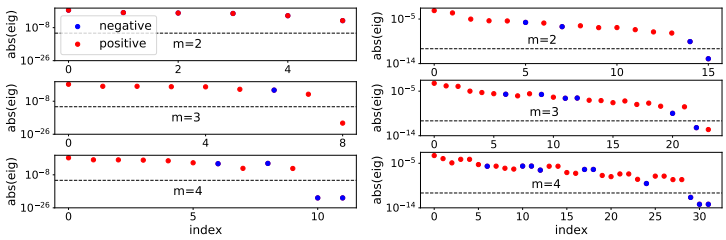
Theorem (degeneracy of embedded critical points, informal Theorem 2). *If output weights of neurons in each layer of a DNN at a critical point* $\boldsymbol{\theta}_{\text{narr}}$ *are not all zero, then, for any* K*-neuron wider DNN,* $\boldsymbol{\theta}_{\text{narr}}$ *can be embedded to a* K*-dimensional critical hyperplane.*

Remark. *By above theorem, each step of embedding of a critical point in general is accompanied by an increased degree of degeneracy. Therefore, degenerate critical points in general widely exist in the loss landscape of a DNN, and non-degenerate critical points are rare because they often become degenerate once embedded to a wider DNN.*

In previous studies, degeneracy is often considered as a consequence of over-parameterization depending on the size of training data n. Specifically, Cooper (2021) proves that the degree of degeneracy of global minima is $m - n$ for 1-d output, where m is the number of network parameters. However, we demonstrate by the above theorem that the degeneracy of critical points is actually a key characteristic of loss landscape of NNs independent of the size of training data, which is very different from loss landscape of conventional problems like protein folding.

3.2 Numerical experiments

Increment of degeneracy through embedding. We train a two-layer NN of width $m_{\text{small}} = 2$ to learn data of Fig. 1 shown in Fig. 3(a) or Iris dataset (Fisher, 1936) in Fig. 3(b) to a critical point. We then embed this critical point to networks of width $m = 3$ and $m = 4$ through a one-step or a two-step embedding, respectively. It is obvious from Fig. 3 that each step of embedding incurs one more zero eigenvalue in the Hessian matrix, which conforms with Theorem 3.1. Moreover, in Fig. 3(a), for $m = 2$, all eigenvalues are positive (red) indicating the critical point obtained by training is a local or global minimum. After embedding, this point becomes a saddle due to the emergence of negative eigenvalues (blue). Specifically, in both Fig. 3(a) and (b), we observe a steady increase of significant negative eigenvalues, e.g., from 0 to 1 to 2 in (a) and from 3 to 5 to 7 in (b), which implies reduced difficulty in escaping from the corresponding critical point in a wider NN during the training.



(a) synthetic data (b) Iris data

Figure 3: Eigenvalues of Hessian of NNs at the critical points embedded from the NN with width $m_{\text{small}} = 2$ for learning data of Fig. 1 in (a) and for Iris dataset in (b). The value of m in each sub-figure is the NN width after embedding. The auxiliary dash line in each sub-figure is $y = 10^{-11}$.

Diagram of loss landscape. In Fig. 4, we present a diagram of loss landscape of a width-3 two-layer tanh DNN to visualize a set of its critical points predicted by the embedding principle, i.e., critical points embedded from network of width-1 or -2 respectively as well as critical points that cannot be obtained through embedding. Through the training of width-1, -2, -3 network respectively on the training data presented in Fig. 1 for multiple trials, we discover 1 critical point for width-1 network, 2

critical points for width-2 network and 1 critical point for width-3 network that cannot be embedded from a narrower NN. Then, embedding all these four critical points to critical points/hyperplanes of loss landscape of the width-3 network, we obtain four sets of critical points with their loss values, output functions, degrees of degeneracy and width of network they embedded from illustrated in Fig. 4. This diagram immediately tells us what attracts the gradient-based training trajectory for a width-3 network. Specifically, if stagnation happens during the training, this diagram informs us the potential loss values and output functions at stagnation, which could help us better understand the nonlinear training process of not only a width-3 network but also much wider networks due to the embedding principle. Furthermore, as illustrated in Fig. 1 for the training process of a 500-neuron NN, saddle points of a wide NN, effectively local or global minima of narrow NNs, composes a trajectory, which may serve as a compass for achieving a global minimum from narrow NNs of low complexity.

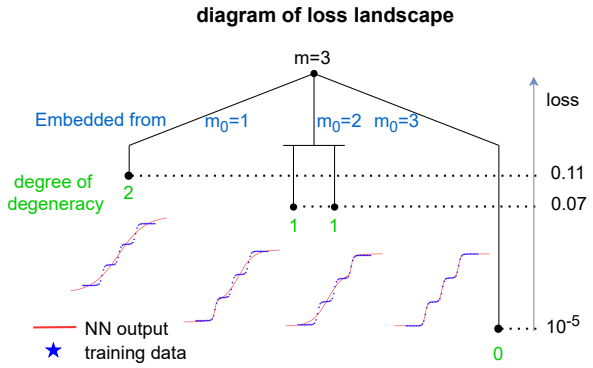


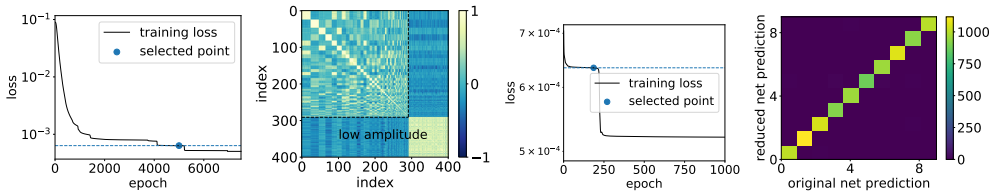
Figure 4: Diagram of loss landscape of a width-3 two-layer tanh NN. Each branch of critical points is shown with their loss values, width of NN they embedded from, degree of degeneracy and output function.

Reduction of DNN at critical point. The embedding principle predicts a class of critical points of a NN embedded from much narrower NNs. At such a critical point, we shall be able to find neuron groups, within which neurons have similar orientation of input weights presumably originated from the same neuron of a narrow NN through embedding. This prediction is confirmed by the following experiment in Fig. 5. We train a width-400 two-layer ReLU NN on 1000 training samples of the MNIST dataset with small initialization. At the blue dot in Fig. 5(a), the loss decreases very slowly, presumably very close to a saddle point. We then examine the orientation similarity between each pair of neuron input weights by computing the inner product of two normalized input weight. As shown in Fig. 5(b), there emerge 58 groups of neurons (neurons with very small amplitudes are neglected and later directly removed), where similarity between input weights in the same group is at least 0.9. For each group S_{similar}, we randomly select a neuron j, replace its output weight by $\sum_{k\in S_{\text{similar}}} a_k \|\boldsymbol{w}_k\|_2 / \|\boldsymbol{w}_j\|_2$, and discard all other neurons in the group. The parameter set before reduction is denoted by $\boldsymbol{\theta}_{\text{ori}}$, and after reduction by $\boldsymbol{\theta}_{\text{redu}}$. Width of the NN is reduced from 400 to 58. We train the reduced NN from $\boldsymbol{\theta}_{\text{redu}}$ as shown in Fig. 5, which stagnated after a few steps at the same loss value as the blue point in Fig. 5(a) marked by the blue dash and represented by the blue point in Fig. 5(c). We then compare the prediction between original model and the reduced model at the corresponding blue points on 10000 test data as shown in Fig.5(d). For each grid, color indicates the frequency of that prediction pair. Specifically, the highlight of diagonal element indicts high prediction agreement of two models (overall $\sim 98.5\%$). Therefore, this critical point of the reduced width-58 NN well matches the critical point of the original width-400 NN, clearly demonstrating the relevance of our embedding principle to real dataset training.

4 Preliminaries

4.1 Deep Neural Networks

Consider L-layer ($L \geq 2$) fully-connected DNNs with a general differentiable activation function. We regard the input as the 0-th layer and the output as the L-th layer. Let m_l be the number of



(a) initial loss (b) orientation similarity (c) loss after reduction (d) prediction similarity

Figure 5: (a) The training loss of the initial network on MNIST. The blue point is selected for reduction. (b) The normalized inner product of input weights for different neurons. The abscissa and ordinate represent neuron index. Neurons in "low amplitude" region has much lower amplitude than others, hence are removed. (c) The training loss of the reduced network. Blue dash indicates the same loss value as the blue dash in (a). The blue point is selected as a representative for comparison. (d) Prediction similarity. For each grid, color indicates the frequency of that prediction pair.

neurons in the l-th layer. In particular, $m_0 = d$ and $m_L = d'$. For any $i, k \in \mathbb{N}$ and $i < k$, we denote $[i : k] = \{i, i+1, \ldots, k\}$. In particular, we denote $[k] := \{1, 2, \ldots, k\}$. Given weights $W^{[l]} \in \mathbb{R}^{m_l \times m_{l-1}}$ and bias $b^{[l]} \in \mathbb{R}^{m_l}$ for $l \in [L]$, we define the collection of parameters $\boldsymbol{\theta}$ as a $2L$-tuple (an ordered list of $2L$ elements) whose elements are matrices or vectors

$$\boldsymbol{\theta} = \left(\boldsymbol{\theta}|_1, \cdots, \boldsymbol{\theta}|_L\right) = \left(\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \ldots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]}\right). \tag{1}$$

where the l-th layer parameters of $\boldsymbol{\theta}$ is the ordered pair $\boldsymbol{\theta}|_l = \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right), \quad l \in [L]$. We may misuse of notation and identify $\boldsymbol{\theta}$ with its vectorization $\mathrm{vec}(\boldsymbol{\theta}) \in \mathbb{R}^M$ with $M = \sum_{l=0}^{L-1}(m_l + 1)m_{l+1}$.

Given $\boldsymbol{\theta} \in \mathbb{R}^M$, the neural network function $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ is defined recursively. First, we write $\boldsymbol{f}_{\boldsymbol{\theta}}^{[0]}(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Then for $l \in [L-1]$, $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}$ is defined recursively as $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}) = \sigma(\boldsymbol{W}^{[l]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]}(\boldsymbol{x}) + \boldsymbol{b}^{[l]})$. Finally, we denote

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{f}_{\boldsymbol{\theta}}^{[L]}(\boldsymbol{x}) = \boldsymbol{W}^{[L]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}) + \boldsymbol{b}^{[L]}. \tag{2}$$

For notational simplicity, we may drop the subscript $\boldsymbol{\theta}$ in $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}$, $l \in [0 : L]$.

We introduce the following notions for the convenience of the presentation in this paper.

Definition 3 (Wider/narrower DNN). *We write* $\mathrm{NN}(\{m_l\}_{l=0}^L)$ *for a fully-connected neural network with width* $(m_0, \ldots, m_L)$*. Given two L-layer (*$L \geq 2$*) fully-connected neural networks* $\mathrm{NN}(\{m_l\}_{l=0}^L)$ *and* $\mathrm{NN}'(\{m'_l\}_{l=0}^L)$*, if* $m'_0 = m_0$*,* $m'_L = m_L$*, and for any* $l \in [L-1]$*,* $m'_l \geq m_l$ *and* $K = \sum_{l=1}^{L-1}(m'_l - m_l) \in \mathbb{N}_+$*, then we say that* $\mathrm{NN}'(\{m'_l\}_{l=0}^L)$ *is* K*-neuron wider than* $\mathrm{NN}(\{m_l\}_{l=0}^L)$ *and* $\mathrm{NN}(\{m_l\}_{l=0}^L)$ K*-neuron narrower than* $\mathrm{NN}'(\{m'_l\}_{l=0}^L)$*.*

4.2 Loss Function and Training Dynamics

The training data set is denoted as $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^{d'}$. For simplicity, here we assume an unknown function $\boldsymbol{f}^*$ satisfying $\boldsymbol{f}^*(\boldsymbol{x}_i) = \boldsymbol{y}_i$ for $i \in [n]$. The empirical risk reads as

$$R_S(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{f}^*(\boldsymbol{x}_i)) = \mathbb{E}_S \ell(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{f}^*(\boldsymbol{x})). \tag{3}$$

where the expectation $\mathbb{E}_S h(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^n h(\boldsymbol{x}_i)$ for any function $h : \mathbb{R}^d \to \mathbb{R}$ and the loss function $\ell(\cdot, \cdot)$ is differentiable and the derivative of ℓ with respect to its first argument is denoted by $\nabla \ell(\boldsymbol{y}, \boldsymbol{y}^*)$. Generally, we always take derivatives/gradients of ℓ in its first argument with respect to any parameter. We consider gradient flow of R_S as the training dynamics, i.e., $\mathrm{d}\boldsymbol{\theta}/\mathrm{d}t = -\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta})$ with $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$.

We define the error vectors $\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]} = \nabla_{\boldsymbol{f}^{[l]}} \ell$ for $l \in [L]$ and the feature gradients $\boldsymbol{g}_{\boldsymbol{\theta}}^{[L]} = \mathbf{1}$ and $\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]} = \sigma^{(1)}\left(\boldsymbol{W}^{[l]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]} + \boldsymbol{b}^{[l]}\right)$ for $l \in [L-1]$. Here $\sigma^{(1)}$ is the first derivative of σ. We call $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}$,

$l \in [L]$ feature vectors. The collections of feature vectors, feature gradients, and error vectors are $\boldsymbol{F}_{\boldsymbol{\theta}} = \{\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}\}_{l=1}^{L}, \boldsymbol{G}_{\boldsymbol{\theta}} = \{\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]}\}_{l=1}^{L}, \boldsymbol{Z}_{\boldsymbol{\theta}} = \{\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]}\}_{l=1}^{L}$. Using backpropagation, we can calculate the gradients as follows

$$\boldsymbol{z}_{\boldsymbol{\theta}}^{[L]} = \nabla \ell, \quad \boldsymbol{z}_{\boldsymbol{\theta}}^{[l]} = (\boldsymbol{W}^{[l+1]})^{\intercal} \boldsymbol{z}_{\boldsymbol{\theta}}^{[l+1]} \circ \boldsymbol{g}_{\boldsymbol{\theta}}^{[l+1]}, \quad l \in [L-1],$$
$$\nabla_{\boldsymbol{W}^{[l]}} \ell = \boldsymbol{z}_{\boldsymbol{\theta}}^{[l]} \circ \boldsymbol{g}_{\boldsymbol{\theta}}^{[l]} (\boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]})^{\intercal}, \quad \nabla_{\boldsymbol{b}^{[l]}} \ell = \boldsymbol{z}_{\boldsymbol{\theta}}^{[l]} \circ \boldsymbol{g}_{\boldsymbol{\theta}}^{[l]}, \quad l \in [L].$$

Here we use $\circ$ for the Hadamard product of two matrices of the same dimension.

5 Critical embedding

We introduce the one-step embedding for the DNNs which will lead us to general embeddings.

Definition 4 (one-step embedding). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \ldots, m_L)$ and network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]}) \in \mathbb{R}^M$, for any $l \in [L-1]$ and any $s \in [m_l]$, we define the linear operators $\mathcal{T}_{l,s}$ and $\mathcal{V}_{l,s}$ applying on $\boldsymbol{\theta}$ as follows*

$$\begin{aligned}
\mathcal{T}_{l,s}(\boldsymbol{\theta})|_k &= \boldsymbol{\theta}|_k, \quad k \neq l, l+1, \\
\mathcal{T}_{l,s}(\boldsymbol{\theta})|_l &= \left(\begin{bmatrix} \boldsymbol{W}^{[l]} \\ \boldsymbol{W}_{s,[1:m_{l-1}]}^{[l]} \end{bmatrix}, \begin{bmatrix} \boldsymbol{b}^{[l]} \\ \boldsymbol{b}_s^{[l]} \end{bmatrix} \right), \quad \mathcal{T}_{l,s}(\boldsymbol{\theta})|_{l+1} = \left(\left[\boldsymbol{W}^{[l+1]}, \boldsymbol{0}_{m_{l+1} \times 1} \right], \boldsymbol{b}^{[l+1]} \right), \\
\mathcal{V}_{l,s}(\boldsymbol{\theta})|_k &= \left(\boldsymbol{0}_{m_k \times m_{k-1}}, \boldsymbol{0}_{m_k \times 1} \right), \quad k \neq l, l+1, \\
\mathcal{V}_{l,s}(\boldsymbol{\theta})|_l &= \left(\boldsymbol{0}_{(m_l+1) \times m_{l-1}}, \boldsymbol{0}_{(m_l+1) \times 1} \right), \\
\mathcal{V}_{l,s}(\boldsymbol{\theta})|_{l+1} &= \left(\left[\boldsymbol{0}_{m_{l+1} \times (s-1)}, -\boldsymbol{W}_{[1:m_{l+1}],s}^{[l+1]}, \boldsymbol{0}_{m_{l+1} \times (m_l - s)}, \boldsymbol{W}_{[1:m_{l+1}],s}^{[l+1]} \right], \boldsymbol{0}_{m_{l+1} \times 1} \right).
\end{aligned}$$

Then the one-step embedding operator $\mathcal{T}_{l,s}^{\alpha}$ is defined as for any $\boldsymbol{\theta} \in \mathbb{R}^M$

$$\mathcal{T}_{l,s}^{\alpha}(\boldsymbol{\theta}) = (\mathcal{T}_{l,s} + \alpha \mathcal{V}_{l,s})(\boldsymbol{\theta}).$$

Note that the resulting parameter $\mathcal{T}_{l,s}^{\alpha}(\boldsymbol{\theta})$ corresponds to a L-layer fully-connected neural network with width $(m_0, \ldots, m_{l-1}, m_l + 1, m_{l+1}, \ldots, m_L)$.

An illustration of $\mathcal{T}_{l,s}$, $\mathcal{V}_{l,s}$, and $\mathcal{T}_{l,s}^{\alpha}$ can be found in Fig. 6 in Appendix.

Lemma 1. *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \ldots, m_L)$, for any network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ and for any $l \in [L-1]$, $s \in [m_l]$, we have the expressions for $\boldsymbol{\theta}' := \mathcal{T}_{l,s}^{\alpha}(\boldsymbol{\theta})$*

(i) feature vectors in $\boldsymbol{F}_{\boldsymbol{\theta}'}$: $\boldsymbol{f}_{\boldsymbol{\theta}'}^{[l']} = \boldsymbol{f}_{\boldsymbol{\theta}'}^{[l']}$, $l' \neq l$ and $\boldsymbol{f}_{\boldsymbol{\theta}'}^{[l]} = \left[(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]})^{\intercal}, (\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]})_s \right]^{\intercal}$;

(ii) feature gradients in $\boldsymbol{G}_{\boldsymbol{\theta}'}$: $\boldsymbol{g}_{\boldsymbol{\theta}'}^{[l']} = \boldsymbol{g}_{\boldsymbol{\theta}}^{[l']}$, $l' \neq l$ and $\boldsymbol{g}_{\boldsymbol{\theta}'}^{[l]} = \left[(\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]})^{\intercal}, (\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]})_s \right]^{\intercal}$;

(iii) error vectors in $\boldsymbol{Z}_{\boldsymbol{\theta}'}$: $\boldsymbol{z}_{\boldsymbol{\theta}'}^{[l']} = \boldsymbol{z}_{\boldsymbol{\theta}}^{[l']}$, $l' \neq l$ and $\boldsymbol{z}_{\boldsymbol{\theta}'}^{[l]} = \left[(\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]})_{[1:s-1]}^{\intercal}, (1-\alpha)(\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]})_s, (\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]})_{[s+1:m_l]}^{\intercal}, \alpha(\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]})_s \right]^{\intercal}$.

An illustration of $\boldsymbol{F}_{\boldsymbol{\theta}}$ and $\boldsymbol{Z}_{\boldsymbol{\theta}}$ can be found in Fig. 7 in Appendix.

Proposition 1 (one-step embedding preserves network properties). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \ldots, m_L)$, for any network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ and for any $l \in [L-1]$, $s \in [m_l]$, the following network properties are preserved for $\boldsymbol{\theta}' = \mathcal{T}_{l,s}^{\alpha}(\boldsymbol{\theta})$:*

(i) output function is preserved: $f_{\boldsymbol{\theta}'}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for all $\boldsymbol{x}$;

(ii) empirical risk is preserved: $R_S(\boldsymbol{\theta}') = R_S(\boldsymbol{\theta})$;

(iii) the sets of features are preserved: $\left\{ \left(\boldsymbol{f}_{\boldsymbol{\theta}'}^{[l]} \right)_i \right\}_{i \in [m_l+1]} = \left\{ \left(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]} \right)_i \right\}_{i \in [m_l]}$ and $\left\{ \left(\boldsymbol{f}_{\boldsymbol{\theta}'}^{[l']} \right)_i \right\}_{i \in [m_{l'}]} = \left\{ \left(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l']} \right)_i \right\}_{i \in [m_{l'}]}$ for $l' \in [L] \backslash \{l\}$;

Theorem 1 (**criticality preserving**). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \ldots, m_L)$, for any network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ and for any $l \in [L-1]$, $s \in [m_l]$, if $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}) = \mathbf{0}$, then $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}') = \mathbf{0}$.*

Lemma 2 (**increment of the degree of degeneracy**). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \ldots, m_L)$, if there exists $l \in [L-1]$, $s \in [m_l]$, and a d-dimensional manifold $\mathcal{M}$ consisting of critical points of R_S such that for any $\boldsymbol{\theta} \in \mathcal{M}$, $\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \neq \mathbf{0}$, then $\mathcal{M}' := \{\mathcal{T}^{\alpha}_{l,s}(\boldsymbol{\theta}) | \boldsymbol{\theta} \in \mathcal{M}, \alpha \in \mathbb{R}\}$ is a $(d+1)$-dimensional manifold consists of critical points for the corresponding L-layer fully-connected neural network with width $(m_0, \ldots, m_{l-1}, m_l + 1, m_{l+1}, \ldots, m_L)$.*

Theorem 2 (**degeneracy of embedded critical points**). *Consider two L-layer ($L \geq 2$) fully-connected neural networks $\mathrm{NN}_A(\{m_l\}_{l=0}^L)$ and $\mathrm{NN}_B(\{m'_l\}_{l=0}^L)$ which is K-neuron wider than NN_A. Suppose that the parameters $\boldsymbol{\theta}_A = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ satisfy $\boldsymbol{W}^{[l]} \neq \mathbf{0}$ for each layer $l \in [L]$. Then the parameters $\boldsymbol{\theta}_A$ of NN_A can be critically embedded to a K-dimensional critical hyperplane $\mathcal{M}_B = \{\boldsymbol{\theta}_B + \sum_{i=1}^K \alpha_i \boldsymbol{v}_i | \alpha_i \in \mathbb{R}\}$ of loss landscape of NN_B. Here $\boldsymbol{\theta}_B = (\prod_{i=1}^K \mathcal{T}_{l_i,s_i})(\boldsymbol{\theta}_A)$ and $\boldsymbol{v}_i = \mathcal{T}_{l_K,s_K} \cdots \mathcal{V}_{l_i,s_i} \cdots \mathcal{T}_{l_1,s_1} \boldsymbol{\theta}_A$.*

Note that neuron-index permutation among the same layer is a trivial criticality invariant transform. More discussions about it, specifically for NNs of homogeneous activations like ReLU, can be found in Section A.1 in Appendix.

6 Conclusion and discussion

In this work, we prove an embedding principle that loss landscape of a DNN *contains* all critical points of all the narrower DNNs. This embedding principle unravels wide existence of highly degenerate critical points with low complexity in the loss landscape of a wide DNN, i.e., critical points with low-complexity output function and degenerate Hessian matrix, embedded from critical points of narrow DNNs. With such a loss landscape of DNN, the gradient-based training has the potential of getting attracted or even converging to a low complexity critical point as confirmed by above numerical experiments, which implies a potential implicit regularization towards low-complexity function of nonlinear DNN training dynamics.

Moreover, through critical embedding, a critical point in form of a common non-degenerate local minimum of a narrow DNN not only becomes degenerate in general, but also may become a saddle point as illustrated by numerical experiments. This may be the reason underlying the general easy optimization of wide DNNs observed in practice even beyond the linear/kernel/NTK regime. We will perform more detailed analysis as well as numerical experiments specifically about this minimum-to-saddle transition later.

At the essence, the embedding principle results from the layer structure of a neural network model, which allows arbitrary neuron addition, input weight copying and output weight splitting within each layer. Therefore, though results in this work assume fully-connected NNs, these can be easily extended to other DNN architectures. Currently, depth serves as a preset hyperparameter in our analysis. Whether loss landscape of DNNs of different depth has certain embedding relation for specific DNN architectures such as ResNet is an interesting open problem.

We remark that our embedding principle applies for landscape of general loss functions. Although for loss functions like cross entropy, meaningful finite critical point may not exist as its parameters diverge in general throughout the training, yet it is reasonable to expect that critical embedding may provide us certain approximate critical points from narrow NNs. Of course, how to properly define an approximate critical point is in itself a problem of interest. And we leave this problem for the future study.

Overall, our embedding principle provides the first clear picture about the general structure of critical points of DNN loss landscape, which is fundamental to the theoretical understanding of both training and generalization behavior of DNNs as well as the design of optimization algorithms. Of course, the study of loss landscape of DNN is far from complete. This work serves as a starting point for a novel line of research, which finally leads to an exact and comprehensive theoretical description about loss landscape of DNNs as well as an understanding of its profound impact on training and generalization.

References

W. E, C. Ma, S. Wojtowytsch, L. Wu, Towards a mathematical understanding of neural network-based machine learning: What we know and what we don't, arXiv preprint arXiv:2009.10713 (2020).

I. Skorokhodov, M. Burtsev, Loss landscape sightseeing with multi-point optimization, arXiv preprint arXiv:1910.03867 (2019).

L. Breiman, Reflections after refereeing papers for nips, The Mathematics of Generalization XX (1995) 11–15.

C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning requires rethinking generalization, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.

D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, S. Lacoste-Julien, A closer look at memorization in deep networks, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 233–242.

D. Kalimeris, G. Kaplun, P. Nakkiran, B. L. Edelman, T. Yang, B. Barak, H. Zhang, SGD on neural networks learns functions of increasing complexity, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 3491–3501.

P. Jin, L. Lu, Y. Tang, G. E. Karniadakis, Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness, Neural Networks 130 (2020) 85–99.

Z.-Q. J. Xu, Y. Zhang, Y. Xiao, Training behavior of deep neural network in frequency domain, International Conference on Neural Information Processing (2019) 264–274.

Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, Z. Ma, Frequency principle: Fourier analysis sheds light on deep neural networks, Communications in Computational Physics 28 (2020) 1746–1767.

N. Rahaman, D. Arpit, A. Baratin, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of deep neural networks, International Conference on Machine Learning (2019).

T. Luo, Z.-Q. J. Xu, Z. Ma, Y. Zhang, Phase diagram for two-layer relu neural networks at infinite-width limit, Journal of Machine Learning Research 22 (2021) 1–47.

L. Chizat, F. R. Bach, On the global convergence of gradient descent for over-parameterized models using optimal transport, in: NeurIPS 2018, 2018.

C. Ma, L. Wu, W. E, The quenching-activation behavior of the gradient descent dynamics for two-layer neural network models, arXiv preprint arXiv:2006.14450 (2020).

N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, M. Smelyanskiy, On large-batch training for deep learning: Generalization gap and sharp minima, in: 5th International Conference on Learning Representations, ICLR 2017, 2017.

L. Wu, Z. Zhu, et al., Towards understanding generalization of deep learning: Perspective of loss landscapes, arXiv preprint arXiv:1706.10239 (2017).

H. He, G. Huang, Y. Yuan, Asymmetric valleys: Beyond sharp and flat local minima, arXiv preprint arXiv:1902.00744 (2019).

Y. Cooper, Global minima of overparameterized neural networks, SIAM Journal on Mathematics of Data Science 3 (2021) 676–691. doi:10.1137/19M1308943.

L. Sagun, L. Bottou, Y. LeCun, Singularity of the hessian in deep learning, arXiv preprint arXiv:1611.07476 (2016).

A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, Y. LeCun, The loss surfaces of multilayer networks, in: Artificial intelligence and statistics, PMLR, 2015, pp. 192–204.

A. Jacot, C. Hongler, F. Gabriel, Neural tangent kernel: Convergence and generalization in neural networks, in: NeurIPS 2018, 2018.

S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, R. Wang, On exact computation with an infinitely wide neural net, in: NeurIPS 2019, 2019.

Y. Zhang, Z.-Q. J. Xu, T. Luo, Z. Ma, A type of generalization error induced by initialization in deep neural networks, in: MSML 2020, 2020.

S. Du, J. Lee, H. Li, L. Wang, X. Zhai, Gradient descent finds global minima of deep neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 1675–1685.

D. Zou, Y. Cao, D. Zhou, Q. Gu, Stochastic gradient descent optimizes over-parameterized deep relu networks, arXiv preprint arXiv:1811.08888 (2018).

Z. Allen-Zhu, Y. Li, Z. Song, A convergence theory for deep learning via over-parameterization, in: International Conference on Machine Learning, PMLR, 2019, pp. 242–252.

W. E, C. Ma, L. Wu, A comparative analysis of the optimization and generalization property of two-layer neural network and random feature models under gradient descent dynamics, arXiv preprint arXiv:1904.04326 (2019).

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

G. M. Rotskoff, E. Vanden-Eijnden, Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks, in: NeurIPS 2018, 2018.

J. Sirignano, K. Spiliopoulos, Mean field analysis of neural networks: A central limit theorem, Stochastic Processes and their Applications 130 (2020) 1820–1852.

S. Goldt, M. Mézard, F. Krzakala, L. Zdeborová, Modeling the influence of data structure on learning in neural networks: The hidden manifold model, Physical Review X 10 (2020) 041044.

S. He, X. Wang, S. Shi, M. R. Lyu, Z. Tu, Assessing the bilingual knowledge learned by neural machine translation models, arXiv preprint arXiv:2004.13270 (2020).

C. Mingard, J. Skalse, G. Valle-Pérez, D. Martínez-Rubio, V. Mikulik, A. A. Louis, Neural networks are a priori biased towards boolean functions with low entropy, arXiv preprint arXiv:1909.11522 (2019).

R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of eugenics 7 (1936) 179–188.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 5.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3.2

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] the provider information Will be shown in Acknowledgement.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [No] The MNIST datatset is well known.

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

Lemma (Lemma 1)**.** *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \dots, m_L)$, for any network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ and for any $l \in [L-1]$, $s \in [m_l]$, we have the expressions for $\boldsymbol{\theta}' := \mathcal{T}^{\alpha}_{l,s}(\boldsymbol{\theta})$ (see Fig. 7 for an illustration)*

(i) feature vectors in $\boldsymbol{F}_{\boldsymbol{\theta}'}$: $\boldsymbol{f}^{[l']}_{\boldsymbol{\theta}'} = \boldsymbol{f}^{[l']}_{\boldsymbol{\theta}}$, $l' \neq l$ and $\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'} = \left[(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})^{\intercal}, (\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})_s\right]^{\intercal}$;

(ii) feature gradients in $\boldsymbol{G}_{\boldsymbol{\theta}'}$: $\boldsymbol{g}^{[l']}_{\boldsymbol{\theta}'} = \boldsymbol{g}^{[l']}_{\boldsymbol{\theta}}$, $l' \neq l$ and $\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}'} = \left[(\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}})^{\intercal}, (\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}})_s\right]^{\intercal}$;

(iii) error vectors in $\boldsymbol{Z}_{\boldsymbol{\theta}'}$: $\boldsymbol{z}^{[l']}_{\boldsymbol{\theta}'} = \boldsymbol{z}^{[l']}_{\boldsymbol{\theta}}$, $l' \neq l$
and $\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}'} = \left[(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})^{\intercal}_{[1:s-1]}, (1-\alpha)(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s, (\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})^{\intercal}_{[s+1:m_l]}, \alpha(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s\right]^{\intercal}$.

Proof. (i) By the construction of $\boldsymbol{\theta}'$, it is clear that $\boldsymbol{f}^{[l']}_{\boldsymbol{\theta}'} = \boldsymbol{f}^{[l']}_{\boldsymbol{\theta}}$ for any $l' \in [l-1]$. Then

$$\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'} = \sigma \circ \left(\begin{bmatrix} \boldsymbol{W}^{[l]} \\ \boldsymbol{W}^{[l]}_{s,[1:m_{l-1}]} \end{bmatrix} \boldsymbol{f}^{[l-1]}_{\boldsymbol{\theta}} + \begin{bmatrix} \boldsymbol{b}^{[l]} \\ \boldsymbol{b}^{[l]}_s \end{bmatrix} \right) = \begin{bmatrix} \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}} \\ (\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})_s \end{bmatrix}. \tag{4}$$

Note that

$$\alpha \left[\boldsymbol{0}_{m_{l+1} \times (s-1)}, -\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s}, \boldsymbol{0}_{m_{l+1} \times (m_l - s)}, \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \right] \begin{bmatrix} \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}} \\ (\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})_s \end{bmatrix} = \boldsymbol{0}_{m_{l+1} \times 1}.$$

Thus

$$\boldsymbol{f}^{[l+1]}_{\boldsymbol{\theta}'} = \sigma \circ \left(\left[\boldsymbol{W}^{[m_{l+1}]}, \boldsymbol{0}_{m_{l+1} \times 1} \right] \begin{bmatrix} \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}} \\ (\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})_s \end{bmatrix} + \boldsymbol{0}_{m_{l+1} \times 1} + \boldsymbol{b}^{[l+1]} \right) = \boldsymbol{f}^{[l+1]}_{\boldsymbol{\theta}}. \tag{5}$$

Next, by the construction of $\boldsymbol{\theta}'$ again, it is clear that $\boldsymbol{f}^{[l']}_{\boldsymbol{\theta}'} = \boldsymbol{f}^{[l']}_{\boldsymbol{\theta}}$ for any $l' \in [l+1:L]$.

(ii) The results for feature gradients $\boldsymbol{g}^{[l']}_{\boldsymbol{\theta}'}$, $l' \in [L]$ can be calculated in a similar way.

(iii) By the backpropagation and the above facts in (i), we have $\boldsymbol{z}^{[L]}_{\boldsymbol{\theta}'} = \nabla \ell(\boldsymbol{f}^{[L]}_{\boldsymbol{\theta}'}, \boldsymbol{f}^*) = \nabla \ell(\boldsymbol{f}^{[L]}_{\boldsymbol{\theta}}, \boldsymbol{f}^*) = \boldsymbol{z}^{[L]}_{\boldsymbol{\theta}}$.

Recalling the recurrence relation for $l' \in [l+1:L-1]$, then we recursively obtain the following equality for l' from $L-1$ down to $l+1$:

$$\boldsymbol{z}^{[l']}_{\boldsymbol{\theta}'} = (\boldsymbol{W}^{[l'+1]})^{\intercal} \boldsymbol{z}^{[l'+1]}_{\boldsymbol{\theta}'} \circ \boldsymbol{g}^{[l'+1]}_{\boldsymbol{\theta}'} = (\boldsymbol{W}^{[l'+1]})^{\intercal} \boldsymbol{z}^{[l'+1]}_{\boldsymbol{\theta}} \circ \boldsymbol{g}^{[l'+1]}_{\boldsymbol{\theta}} = \boldsymbol{z}^{[l']}_{\boldsymbol{\theta}}. \tag{6}$$

Next,

$$\begin{aligned} \boldsymbol{z}^{[l]}_{\boldsymbol{\theta}'} &= \left(\left[\boldsymbol{W}^{[m_{l+1}]}, \boldsymbol{0}_{m_{l+1} \times 1} \right] + \alpha \left[\boldsymbol{0}_{m_{l+1} \times (s-1)}, -\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s}, \boldsymbol{0}_{m_{l+1} \times (m_l - s)}, \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \right] \right)^{\intercal} \boldsymbol{z}^{[l+1]}_{\boldsymbol{\theta}} \circ \boldsymbol{g}^{[l+1]}_{\boldsymbol{\theta}} \\ &= \begin{bmatrix} \boldsymbol{z}^{[l]}_{\boldsymbol{\theta}} \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{m_{l+1} \times (s-1)} \\ -\alpha(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s \\ \boldsymbol{0}_{m_{l+1} \times (m_l - s)} \\ \alpha(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s \end{bmatrix} \\ &= \left[(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})^{\intercal}_{[1:s-1]}, (1-\alpha)(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s, (\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})^{\intercal}_{[s+1:m_l]}, \alpha(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s \right]^{\intercal}. \end{aligned} \tag{7}$$

Finally,

$$\begin{aligned} \boldsymbol{z}^{[l-1]}_{\boldsymbol{\theta}'} &= \left[(\boldsymbol{W}^{[l]})^{\intercal}, (\boldsymbol{W}^{[l]})^{\intercal}_{s,[1:m_{l-1}]} \right] \left(\begin{bmatrix} \boldsymbol{z}^{[l]}_{\boldsymbol{\theta}} \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{m_{l+1} \times (s-1)} \\ -\alpha(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s \\ \boldsymbol{0}_{m_{l+1} \times (m_l - s)} \\ \alpha(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}})_s \end{bmatrix} \right) \circ \begin{bmatrix} \boldsymbol{g}^{[l]}_{\boldsymbol{\theta}} \\ (\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}})_s \end{bmatrix} \\ &= (\boldsymbol{W}^{[l]})^{\intercal} \boldsymbol{z}^{[l]}_{\boldsymbol{\theta}} \circ \boldsymbol{g}^{[l]}_{\boldsymbol{\theta}} + \boldsymbol{0}_{m_{l-1} \times 1} \\ &= \boldsymbol{z}^{[l-1]}_{\boldsymbol{\theta}}. \end{aligned} \tag{8}$$

This with the recurrence relation again leads to $\boldsymbol{z}^{[l']}_{\boldsymbol{\theta}'} = \boldsymbol{z}^{[l']}_{\boldsymbol{\theta}}$ for all $l' \in [1:l-1]$. □

Proposition (Proposition 1: one-step embedding preserves network properties). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \dots, m_L)$, for any network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ and for any $l \in [L-1]$, $s \in [m_l]$, the following network properties are preserved for $\boldsymbol{\theta}' = \mathcal{T}^{\alpha}_{l,s}(\boldsymbol{\theta})$:*

(i) output function is preserved: $f_{\boldsymbol{\theta}'}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for all $\boldsymbol{x}$;

(ii) empirical risk is preserved: $R_S(\boldsymbol{\theta}') = R_S(\boldsymbol{\theta})$;

(iii) the sets of features are preserved: $\left\{\left(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'}\right)_i\right\}_{i\in[m_l+1]} = \left\{\left(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}}\right)_i\right\}_{i\in[m_l]}$ and $\left\{\left(\boldsymbol{f}^{[l']}_{\boldsymbol{\theta}'}\right)_i\right\}_{i\in[m_{l'}]} = \left\{\left(\boldsymbol{f}^{[l']}_{\boldsymbol{\theta}}\right)_i\right\}_{i\in[m_{l'}]}$ for $l' \in [L]\backslash\{l\}$;

Proof. The properties (i)–(iii) are direct consequences of Lemma 1. □

Theorem (Theorem 1: criticality preserving). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \dots, m_L)$, for any network parameters $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ and for any $l \in [L-1]$, $s \in [m_l]$, if $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}) = \mathbf{0}$, then $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}') = \mathbf{0}$.*

Proof. Gradient of loss with respect to network parameters of each layer can be computed from $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{Z}$ as follows

$$\nabla_{\boldsymbol{W}^{[l']}} R_S(\boldsymbol{\theta}) = \nabla_{\boldsymbol{W}^{[l']}} \mathbb{E}_S \ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{f}^*(\boldsymbol{x})) = \mathbb{E}_S\left(\boldsymbol{z}^{[l']}_{\boldsymbol{\theta}} \circ \boldsymbol{g}^{[l']}_{\boldsymbol{\theta}} (\boldsymbol{f}^{[l'-1]}_{\boldsymbol{\theta}})^{\intercal}\right),$$
$$\nabla_{\boldsymbol{b}^{[l']}} R_S(\boldsymbol{\theta}) = \nabla_{\boldsymbol{b}^{[l]}} \mathbb{E}_S \ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{f}^*(\boldsymbol{x})) = \mathbb{E}_S(\boldsymbol{z}^{[l']}_{\boldsymbol{\theta}} \circ \boldsymbol{g}^{[l']}_{\boldsymbol{\theta}}).$$

Then, by Lemma 1, we have $\nabla_{\boldsymbol{W}^{[l']}} R_S(\boldsymbol{\theta}') = \nabla_{\boldsymbol{W}^{[l']}} R_S(\boldsymbol{\theta}) = \mathbf{0}$ for $l' \neq l, l+1$ and $\nabla_{\boldsymbol{b}^{[l']}} R_S(\boldsymbol{\theta}') = \nabla_{\boldsymbol{b}^{[l']}} R_S(\boldsymbol{\theta}) = \mathbf{0}$ for $l' \neq l$. Also, for any $j \in [m_{l+1}]$,$k \in [m_l]$, since $(\boldsymbol{z}^{[l+1]}_{\boldsymbol{\theta}'})_j = (\boldsymbol{z}^{[l+1]}_{\boldsymbol{\theta}})_j$, $(\boldsymbol{g}^{[l+1]}_{\boldsymbol{\theta}'})_j = (\boldsymbol{g}^{[l+1]}_{\boldsymbol{\theta}})_j$, and $(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'})_k = (\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})_k$, $(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'})_{m_l+1} = (\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}})_s$, we obtain

$$\nabla_{\boldsymbol{W}^{[l+1]}_{j,k}} R_S(\boldsymbol{\theta}') = \nabla_{\boldsymbol{W}^{[l+1]}_{j,k}} R_S(\boldsymbol{\theta}) = 0,$$
$$\nabla_{\boldsymbol{W}^{[l+1]}_{j,m_l+1}} R_S(\boldsymbol{\theta}') = \nabla_{\boldsymbol{W}^{[l+1]}_{j,s}} R_S(\boldsymbol{\theta}) = 0.$$

Similarly, for any $j \in [m_l]\backslash\{s\}$,$k \in [m_{l-1}]$, we have

$$\nabla_{\boldsymbol{W}^{[l]}_{j,k}} R_S(\boldsymbol{\theta}') = \nabla_{\boldsymbol{W}^{[l]}_{j,k}} R_S(\boldsymbol{\theta}) = 0,$$
$$\nabla_{\boldsymbol{b}^{[l]}_{j}} R_S(\boldsymbol{\theta}') = \nabla_{\boldsymbol{b}^{[l]}_{j}} R_S(\boldsymbol{\theta}) = 0,$$
$$\nabla_{\boldsymbol{W}^{[l]}_{s,k}} R_S(\boldsymbol{\theta}') = (1-\alpha)\nabla_{\boldsymbol{W}^{[l]}_{s,k}} R_S(\boldsymbol{\theta}) = 0,$$
$$\nabla_{\boldsymbol{W}^{[l]}_{m_l+1,k}} R_S(\boldsymbol{\theta}') = \alpha\nabla_{\boldsymbol{W}^{[l]}_{s,k}} R_S(\boldsymbol{\theta}) = 0,$$
$$\nabla_{\boldsymbol{b}^{[l]}_{s}} R_S(\boldsymbol{\theta}') = (1-\alpha)\nabla_{\boldsymbol{b}^{[l]}_{s}} R_S(\boldsymbol{\theta}) = 0,$$
$$\nabla_{\boldsymbol{b}^{[l]}_{m_l+1}} R_S(\boldsymbol{\theta}') = \alpha\nabla_{\boldsymbol{b}^{[l]}_{s}} R_S(\boldsymbol{\theta}) = 0.$$

Collecting all the above equalities, we have $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}') = \mathbf{0}$. □

Lemma (Lemma 2: increment of the degree of degeneracy). *Given a L-layer ($L \geq 2$) fully-connected neural network with width $(m_0, \dots, m_L)$, if there exists $l \in [L-1]$, $s \in [m_l]$, and a d-dimensional manifold $\mathcal{M}$ consisting of critical points of R_S such that for any $\boldsymbol{\theta} \in \mathcal{M}$, $\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \neq \mathbf{0}$, then $\mathcal{M}' := \{\mathcal{T}^{\alpha}_{l,s}(\boldsymbol{\theta}) | \boldsymbol{\theta} \in \mathcal{M}, \alpha \in \mathbb{R}\}$ is a $(d+1)$-dimensional manifold consists of critical points for the corresponding L-layer fully-connected neural network with width $(m_0, \dots, m_{l-1}, m_l + 1, m_{l+1}, \dots, m_L)$.*

Proof. For any $\boldsymbol{\theta} \in \mathcal{M}$, let $\{\boldsymbol{e}_i(\boldsymbol{\theta})\}_{i=1}^d$ be a basis of its tangent space $T_{\boldsymbol{\theta}}\mathcal{M}$. Then for any $\alpha \in \mathbb{R}$, the tangent space of $\boldsymbol{\theta}' = \mathcal{T}^{\alpha}_{l,s}(\boldsymbol{\theta}) \in \mathcal{M}'$ is spanned by $\{\mathcal{T}_{l,s}(\boldsymbol{e}_1(\boldsymbol{\theta})), \cdots, \mathcal{T}_{l,s}(\boldsymbol{e}_d(\boldsymbol{\theta})), \mathcal{V}_{l,s}(\boldsymbol{\theta})\}$. Since $\mathcal{T}_{l,s}$ is linear and injective, $\{\mathcal{T}_{l,s}(\boldsymbol{e}_i(\boldsymbol{\theta}))\}_{i=1}^d$ is also a linearly independent set. Moreover, since $\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],m_l+1} = \mathbf{0}$ for any vector in parameter space applied with $\mathcal{T}_{l,s}$, then,

$\{\mathcal{T}_{l,s}(\boldsymbol{e}_1(\boldsymbol{\theta})), \cdots, \mathcal{T}_{l,s}(\boldsymbol{e}_d(\boldsymbol{\theta})), \mathcal{V}_{l,s}(\boldsymbol{\theta})\}$ are independent if and only if $\mathcal{V}_{l,s}(\boldsymbol{\theta}) \neq \mathbf{0}$, i.e., $\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \neq \mathbf{0}$. □

Theorem (**Theorem 2: degeneracy of embedded critical points**). *Consider two L-layer ($L \geq 2$) fully-connected neural networks $\mathrm{NN}_A(\{m_l\}_{l=0}^L)$ and $\mathrm{NN}_B(\{m'_l\}_{l=0}^L)$ which is K-neuron wider than NN_A. Suppose that the parameters $\boldsymbol{\theta}_A = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ satisfy $\boldsymbol{W}^{[l]} \neq \mathbf{0}$ for each layer $l \in [L]$. Then the parameters $\boldsymbol{\theta}_A$ of NN_A can be critically embedded to a K-dimensional critical hyperplane $\mathcal{M}_B = \{\boldsymbol{\theta}_B + \sum_{i=1}^K \alpha_i \boldsymbol{v}_i | \alpha_i \in \mathbb{R}\}$ of loss landscape of NN_B. Here $\boldsymbol{\theta}_B = (\prod_{i=1}^K \mathcal{T}_{l_i,s_i})(\boldsymbol{\theta}_A)$ and $\boldsymbol{v}_i = \mathcal{T}_{l_K,s_K} \cdots \mathcal{V}_{l_i,s_i} \cdots \mathcal{T}_{l_1,s_1}\boldsymbol{\theta}_A$.*

Proof. The assumption $\boldsymbol{W}^{[l]} \neq \mathbf{0}$, $l \in [L]$ implies the existence of non-silent neurons, i.e., existing $s \in [m_l]$ such that $\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \neq \mathbf{0}$, for any $l \in [L-1]$ with $m'_l > m_l$.

In this proof, we misuse notation and denote $m_l = m_l(\boldsymbol{\theta})$ for the width of the l-th layer for any fully-connected neural network with parameters $\boldsymbol{\theta}$. For such a general network with parameters $\boldsymbol{\theta}$, we introduce the following operator. Given an index set J and for any $l \in [L]$, $s \in [m_l]$, we define

$$\mathcal{V}_{l,s,J}(\boldsymbol{\theta}) = \left(\mathbf{0}_{m_0 \times m_1}, \cdots, \left[\mathbf{0}_{m_{l+1} \times (s-1)}, -\sum_{j \in J} \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],j}, \mathbf{0}_{m_{l+1} \times (m_l - s)}, \sum_{j \in J} \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],j} \right], \cdots \right).$$

Clearly, $\mathcal{V}_{l,s,J} = \sum_{j \in J} \mathcal{V}_{l,s,\{j\}}$. If for all $j \in J$, $\boldsymbol{W}^{[l]}_{j,[1:m_{l-1}]} = \boldsymbol{W}^{[l]}_{s,[1:m_{l-1}]}$, $\boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],j} = \beta_j \sum_{j' \in J} \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],j'}$ and $\sum_{j' \in J} \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],j'} \neq \mathbf{0}$, then for $\beta_s \neq 0$, we have

$$\begin{aligned}
\mathcal{T}^{\alpha}_{l,s,J}\boldsymbol{\theta} &= (\mathcal{T}_{l,s} + \alpha \mathcal{V}_{l,s,J})\boldsymbol{\theta} \\
&= (\mathcal{T}_{l,s} + \alpha \sum_{j \in J} \mathcal{V}_{l,s,j})\boldsymbol{\theta} \\
&= (\mathcal{T}_{l,s} + \alpha \sum_{j \in J} \frac{\beta_j}{\beta_s} \mathcal{V}_{l,s})\boldsymbol{\theta} \\
&= \mathcal{T}^{\alpha'}_{l,s}\boldsymbol{\theta},
\end{aligned}$$

where $\alpha' = \alpha \sum_{j \in J} \frac{\beta_j}{1-\beta_j}$, is simply the one-step critical embedding. We can extend this to the case of $\beta_s = 0$.

Then, for $J' = J \cup \{m_l + 1\}$, $l' = l$, $s' \in J$,

$$\begin{aligned}
\mathcal{V}_{l',s',J'}\mathcal{V}_{l,s,J} &= (\mathcal{V}_{l,s',J} + \mathcal{V}_{l,s',m_l+1})\mathcal{V}_{l,s,J} \\
&= \mathcal{V}_{l,s',J}\mathcal{V}_{l,s,J} + \mathcal{V}_{l,s',m_l+1}\mathcal{V}_{l,s,J} \\
&= \mathcal{V}_{l,s',s}\mathcal{V}_{l,s,J} + \mathcal{V}_{l,s',m_l+1}\mathcal{V}_{l,s,J} \\
&= \mathbf{0}.
\end{aligned}$$

In general, we have

$$\begin{aligned}
\mathcal{V}_{l',s',J'} \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} &= (\mathcal{V}_{l,s',J} + \mathcal{V}_{l,s',m_l+1}) \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} \\
&= \mathcal{V}_{l,s',J} \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} + \mathcal{V}_{l,s',m_l+1} \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} \\
&= \mathcal{V}_{l,s',s} \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} + \mathcal{V}_{l,s',m_l+1} \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} \\
&= \mathbf{0}.
\end{aligned}$$

For $l' \neq l$ or $s' \notin J$, obviously we have $\mathcal{V}_{l',s',J'}\mathcal{V}_{l,s,J} = \mathbf{0}$ and $\mathcal{V}_{l',s',J'} \prod_{i=1}^N \mathcal{T}_{l'_i,s'_i}\mathcal{V}_{l,s,J} = \mathbf{0}$.

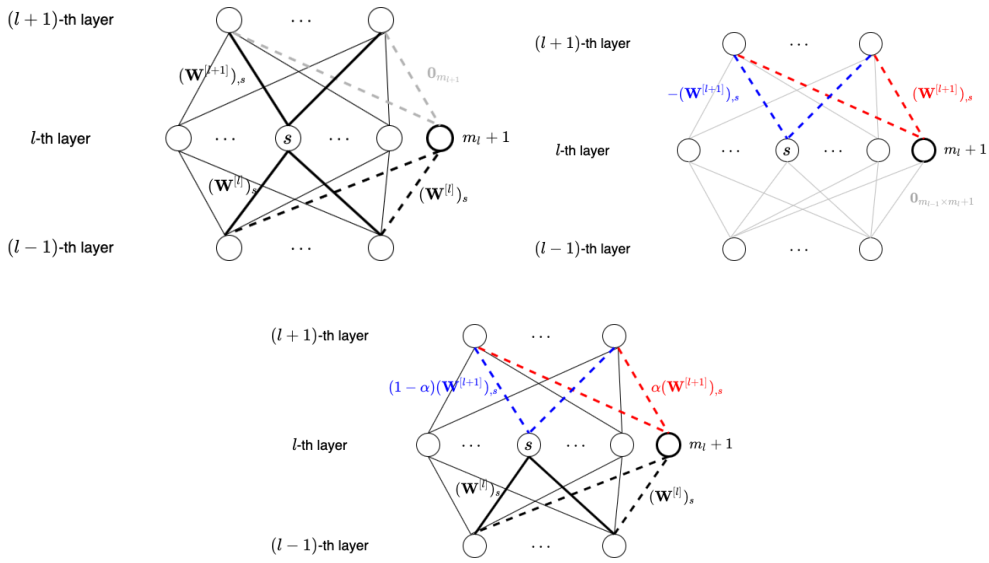


Figure 6: Illustration of $\mathcal{T}_{l,s}$, $\mathcal{V}_{l,s}$, and $\mathcal{T}^{\alpha}_{l,s}$.

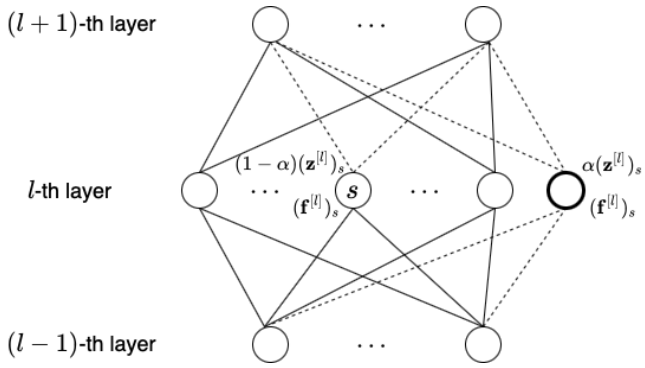


Figure 7: Illustration of $\boldsymbol{F}$ and $\boldsymbol{Z}$

Now we are ready to prove the lemma. Let $J_i = \{s_i\} \cup \{m_l + \#\{i | l_i = l, i \in [j]\} | l_j = l, s_j = s, j \in [K]\}$, where $\#$ indicates number of elements in a set. Then

$$\begin{aligned}
\prod_{i=1}^{K} \mathcal{T}^{\alpha_i}_{l_i,s_i,J_i} &= \prod_{i=1}^{K} (\mathcal{T}_{l_i,s_i} + \alpha_i \mathcal{V}_{l_i,s_i,J_i}) \\
&= \prod_{i=1}^{K} \mathcal{T}_{l_i,s_i} + \sum_{i=1}^{K} \alpha_i \mathcal{T}_{l_K,s_K} \cdots \mathcal{V}_{l_i,s_i,J_i} \cdots \mathcal{T}_{l_1,s_1}, \\
&= \prod_{i=1}^{K} \mathcal{T}_{l_i,s_i} + \sum_{i=1}^{K} \alpha_i \mathcal{T}_{l_K,s_K} \cdots \mathcal{V}_{l_i,s_i} \cdots \mathcal{T}_{l_1,s_1},
\end{aligned}$$

which is a critical embedding for any $[\alpha_i]_{i=1}^{K} \in \mathbb{R}^K$. This completes the proof.

□

A.1 Trivial critical transforms

In general, neuron-index permutation among the same layer is a trivial criticality invariant transform because of the layer-wise intrinsic symmetry of DNN models. Therefore, any critical point/manifold may result in multiple "mirror" critical points/manifolds of the loss landscape through all possible permutations. However, this transform does not inform about the degeneracy of critical points/manifolds.

For any p-homogeneous activation function σ i.e., $\sigma(\beta x) = \beta^p \sigma(x)$ for any $\beta > 0$ and $x \in \mathbb{R}$, we define for any $l \in [L-1]$, $s \in [m_l]$ the following scaling transform $\boldsymbol{\theta}' = \mathcal{S}^{\beta}_{l,s}(\boldsymbol{\theta})$ ($\beta \neq 0$) such that $\boldsymbol{W}'^{[l+1]}_{[1:m_{l+1}],s} = \frac{1}{\beta^p} \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s}$ and $\boldsymbol{W}'^{[l]}_{s,[1:m_{l-1}]} = \beta \boldsymbol{W}^{[l]}_{s,[1:m_{l-1}]}$,$\boldsymbol{b}'^{[l]}_s = \beta \boldsymbol{b}^{[l]}_s$, and all the other entries remains the same. Clearly, this transform is also a critical transform. Moreover, it informs about one more degenerate dimension for each neuron with $\left\| \boldsymbol{W}^{[l+1]}_{[1:m_{l+1}],s} \right\|_2 \left\| \left(\boldsymbol{W}^{[l]\intercal}_{s,[1:m_{l-1}]}, \boldsymbol{b}^{[l]}_s \right)^{\intercal} \right\|_2 \neq 0$. This critical scaling transform is trivial in a sense that it is an obvious result of the cross-layer scaling preserving intrinsic to each DNN of homogeneous activation, not relevant to cross-width landscape similarity between DNNs we focus on.