CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.SO-2023-0020

Vol. **x**, No. x, pp. 1-40
xx 2024

Embedding Principle in Depth for the Loss Landscape Analysis of Deep Neural Networks

Zhiwei Bai[1], Tao Luo[1,2], Zhi-Qin John Xu[1,*] and Yaoyu Zhang[1,3,*]

[1] *School of Mathematical Sciences, Institute of Natural Sciences, MOE-LSC, Shanghai Jiao Tong University, Shanghai 200240, P.R. China.*
[2] *CMA-Shanghai, Shanghai Artificial Intelligence Laboratory, Shanghai 200240, P.R. China.*
[3] *Shanghai Center for Brain Science and Brain-Inspired Technology, Shanghai 200240, P.R. China.*

Received 16 May 2023; Accepted 7 January 2024

Summary. In this work, we delve into the relationship between deep and shallow neural networks (NNs), focusing on the critical points of their loss landscapes. We discover an embedding principle in depth that loss landscape of an NN "contains" all critical points of the loss landscapes for shallower NNs. The key tool for our discovery is the critical lifting that maps any critical point of a network to critical manifolds of any deeper network while preserving the outputs. To investigate the practical implications of this principle, we conduct a series of numerical experiments. The results confirm that deep networks do encounter these lifted critical points during training, leading to similar training dynamics across varying network depths. We provide theoretical and empirical evidence that through the lifting operation, the lifted critical points exhibit increased degeneracy. This principle also provides insights into the optimization benefits of batch normalization and larger datasets, and enables practical applications like network layer pruning. Overall, our discovery of the embedding principle in depth uncovers the depth-wise hierarchical structure of deep learning loss landscape, which serves as a solid foundation for the further study about the role of depth for DNNs.

AMS subject classifications: 68T07
Key words: Deep learning, loss landscape, embedding principle.

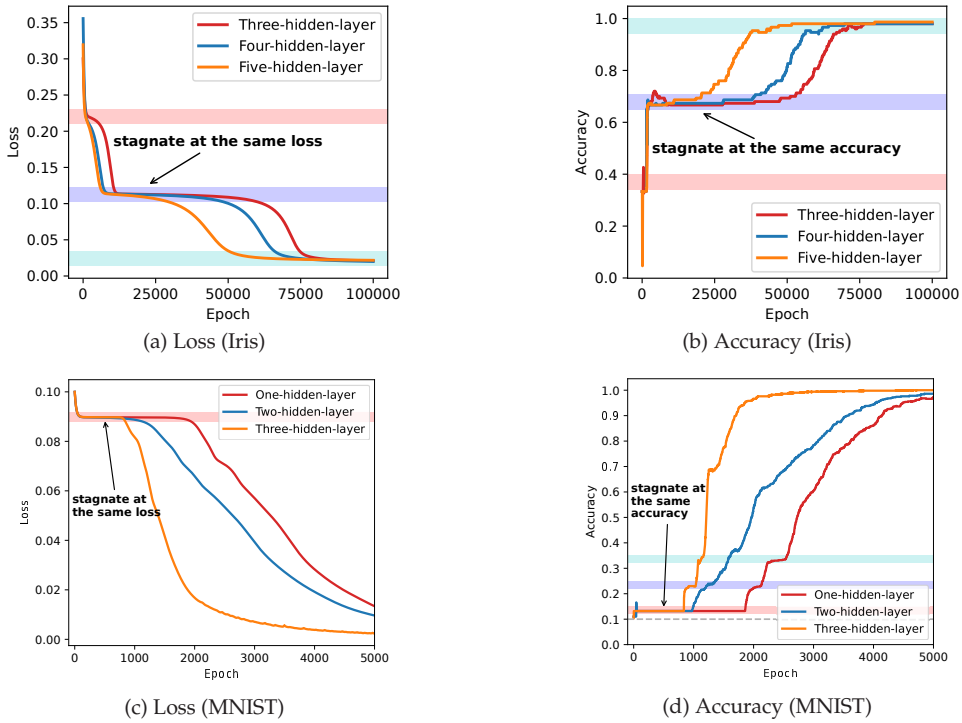
1 Introduction

Deep neural networks (DNNs) have achieved remarkable success in various fields, such as computer vision [18], natural language processing [4], and numerous scientific computing applications [2, 10, 24]. Despite their widespread adoption and empirical achieve-

*Corresponding author. *Email addresses:* bai299@sjtu.edu.cn (Z. Bai), luotao41@sjtu.edu.cn (T. Luo), xuzhiqin@sjtu.edu.cn (Z. Xu), zhyy.sjtu@sjtu.edu.cn (Y. Zhang)

ments, our theoretical understanding of DNNs, particularly regarding their loss landscape and training dynamics, remains limited. The loss landscape of a DNN essentially characterizes the optimization problem encountered during the network's training process. The study of this landscape is of paramount importance as it directly influences not only the efficiency and final outcome of the training process, but also the generalization in overparametered case. Regrettably, the high-dimensionality and non-convex nature of DNNs render their loss landscapes notoriously challenging to comprehend and navigate. The recent discovery of the embedding principle [9,20,30,32] offers insights for analyzing the loss landscape of networks and establishes connections between the loss landscapes of neural networks with varying widths. However, considering the extreme importance of depth for DNNs, it prompts us to question whether a relationship exists between the loss landscapes of networks with different depths. In this paper, we strive to address this fundamental question by conducting a thorough analysis of critical points across varying network depths.

Our theoretical investigation is motivated by the following experimental observations, which hint at the existence of an embedding relationship in depth. As illustrated in Fig. 1, the training of NNs with varying hidden layers, learning the Iris and MNIST



(a) Loss (Iris) (b) Accuracy (Iris)

(c) Loss (MNIST) (d) Accuracy (MNIST)

Figure 1: The training dynamics of networks of different depths exhibit similarity. (a, c) The training loss for NNs of varying depths on the Iris and MNIST datasets, respectively. (b, d) The corresponding training accuracy for NNs of varying depths on the Iris and MNIST datasets, respectively. The color-coded areas indicate periods of slow change in training loss or training accuracy, indicating a possible encounter with a saddle point.

datasets with small initialization and a small learning rate, exhibit a similar behavior. Specifically, in Figs. 1(a) and 1(c), we notice that network trajectories of different depths appear to stagnate at almost the same loss values, with virtually the same training accuracy, as demonstrated in Figs. 1(b) and 1(d). This intriguing observation suggests that the loss landscapes of NNs of varying depths may share a set of critical functions (i.e. output functions of critical points), by which a deep NN can experience a training process similar to that of a shallower one.

Motivated by these observations, we prove in this work an embedding principle in depth for fully-connected NNs, which can be intuitively stated as follows:

Embedding principle in depth: The loss landscape of any network "contains" all critical points of all shallower networks.

Central to our proof of the embedding principle in depth is the introduction of a critical lifting operator. This operator, as proposed in this work, maps any critical point of a shallower NN to critical manifolds (i.e. manifolds consisting of critical points sharing the same loss value) of a target NN, while preserving outputs on the training inputs. Our critical lifting operator predicts a rich class of "simple" critical points, which are derived from shallower NNs and embedded in the loss landscapes of deeper NNs. This thereby explicitly unveils the depth-wise hierarchical structure within the loss landscape of deep learning.

To evaluate the practical implications of the embedding principle in depth, we conduct a comprehensive set of numerical experiments. These experiments reveal that the practical training dynamics of deep NNs indeed encounter these lifted critical points, resulting in similar training dynamics between deep and shallow networks. Furthermore, we observe that through the critical lifting process, lifted critical points exhibit increased degeneracy, which aligns with the empirically observed highly degenerate critical points within the loss landscape [19]. The embedding principle in depth also provides new understanding to the optimization benefits of batch normalization [13] and the use of larger datasets. In the final part of our experimental study, we explore the aspect of network compression, proposing a method for layer pruning.

The remainder of the paper is organized as follows. In Section 2, we review related works. In Section 3, we provide a brief introduction to deep neural networks and the back propagation process. In Section 4, we lay out the theory of the embedding principle in depth. Section 5 presents a range of practical effects to corroborate our theoretical insights. In Section 6, we contrast the differences between the embedding principles in width and depth and discuss other network architectures beyond fully-connected networks. We conclude the paper in Section 7. Detailed proofs are provided in Appendix A.

2 Related works

The loss landscape of deep neural networks is notoriously complex due to its high-dimensionality and non-convex nature [21]. Certain directions of a minimum can exhibit

markedly different sharpness [11]. Moreover, different training algorithms find global minima with different properties, such as SGD often finds a flatter minimum compared with GD [14,25]. Although previous studies have provided detailed investigations on the loss landscape of shallow NNs with specific activations [3,5,22], the relationship between critical points across different network architectures remains largely unexplored.

The recent work [32] introduced an embedding principle (in width) that establishes a relationship between the critical points of a network and its wider counterparts. The principle, which leverages one-step embeddings and their multi-step composition, suggests that the critical points of a network can be embedded into the loss landscape of wider NNs. Similar findings about these composition embeddings have been studied [8, 9,20]. Different from these works studying the effect of width, our work for the first time establishes the embedding relation regarding the extremely important hyperparameter of depth for DNNs.

Using a deeper NN has many advantages. In approximation, a deeper NN has more expressive power [6,7,23]. In optimization, a deeper NN can learn data faster [1,12,28]. In generalization, it has been widely observed that overparameterized deep neural networks often generalize well in practice [29] and a deeper NN may achieve better generalization for real-world problems [12]. Therefore, it is important to understand the effect of depth to the DNN loss landscapes.

The proposed embedding principle in depth suggests a simplicity bias in depth, which is consistent with previous works, for example, the frequency principle [17, 26, 27, 31], which states that DNNs often fit target functions from low to high frequencies during the training, and the block structure [16], which identifies similar representations across many layers in overparameterized networks.

3 Preliminaries

Deep neural networks. Consider a fully connected neural network (NN) with $L\,(L\geq 1)$ layers. Let $i,k\in\mathbb{N}$, and for $i<k$, denote $[i:k]=i,i+1,\ldots,k$. Specifically, denote $[k]:=1,2,\ldots,k$. The input is treated as layer 0 and the output as layer L. The width of layer l is represented by m_l, with $m_0=d$ and $m_L=d'$.

For any parameter $\boldsymbol{\theta}$ of the NN, we consider it as a $2L$-tuple

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

where $\mathbf{W}^{[l]}\in\mathbb{R}^{m_l\times ml-1}$ and $\mathbf{b}^{[l]}\in\mathbb{R}^{ml}$ represent the weight and bias of layer l, respectively. The parameters of layer l in $\boldsymbol{\theta}$ are given as an ordered pair $\boldsymbol{\theta}|_l=(\mathbf{W}^{[l]},\mathbf{b}^{[l]})$ for $l\in[L]$. We may use notation interchangeably 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 the parameter vector $\boldsymbol{\theta}$, the neural network function $f_{\boldsymbol{\theta}}(\cdot)$ can be defined recursively. First, let $f_{\boldsymbol{\theta}}^{[0]}(\boldsymbol{x})=\boldsymbol{x}$ for all $\boldsymbol{x}\in\mathbb{R}^d$. Then for $l\in[L-1]$, $f_{\boldsymbol{\theta}}^{[l]}$ is defined recursively as

$$f_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x})=\sigma\big(\boldsymbol{W}^{[l]}f_{\boldsymbol{\theta}}^{[l-1]}(\boldsymbol{x})+\boldsymbol{b}^{[l]}\big).$$

Finally, we denote

$$f_\theta(x)=f(x;\theta)=f_\theta^{[L]}(x)=W^{[L]}f_\theta^{[L-1]}(x)+b^{[L]}.$$

In the case of residual neural networks (ResNets), if the l-th layer employs a skip connection, then

$$f_\theta^{[l]}(x)=\sigma\big(W^{[l]}f_\theta^{[l-1]}(x)+b^{[l]}\big)+f_\theta^{[l-1]}(x).$$

To enable a comprehensible comparison between deep and shallow networks, we provide a precise definition for the terms "deeper" and "shallower".

Definition 3.1 (Deeper/Shallower)**.** *Given two* NN*s*

$$\mathrm{NN}\big(\{m_l\}_{l=0}^L\big),\quad \mathrm{NN}'\big(\{m_l'\},l\in\{0,1,\ldots,q,\hat{q},q+1,\ldots,L\}\big).$$

If

$$m_1'=m_1,\cdots,m_q'=m_q,m_{\hat{q}}'\geq\min\{m_q,m_{q+1}\},m_{q+1}'=m_{q+1},\cdots,m_L'=m_L,$$

then we say NN′ *is one-layer deeper than* NN*, and conversely,* NN *is one-layer shallower than* NN′*. J-layer deeper (or shallower) is defined by the composition of one-layer deeper (or shallower).*

Remark 3.1. If an NN is termed "deeper" than another NN, it signifies that the former can be derived by incorporating additional layers of adequate widths to the latter. Note that, for the sake of notational convenience, the layer index l for the deeper NN is utilized as a placeholder index, adhering to a specific order of $\{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\}$.

Loss function. We designate the training data and training inputs as $S=\{(x_i,y_i)\}_{i=1}^n$ and $S_x=\{x_i\}_{i=1}^n$, respectively, where $x_i\in\mathbb{R}^d$ and $y_i\in\mathbb{R}^{d'}$. For the sake of convenience, we presuppose an unknown function f^* such that $f^*(x_i)=y_i$ holds for $i\in[n]$. The empirical risk can be expressed as

$$R_S(\theta)=\frac{1}{n}\sum_{i=1}^n\ell\big(f(x_i,\theta),f^*(x_i)\big)=\mathbb{E}_S\ell\big(f(x,\theta),f^*(x)\big),$$

where the expectation $\mathbb{E}_Sh(x):=(\sum_{i=1}^n h(x_i))/n$ is defined for any function $h:\mathbb{R}^d\to\mathbb{R}$. We denote the derivative of the loss function ℓ with respect to its first argument as $\nabla\ell(y,y^*)$. The training dynamics are treated as the gradient flow of $R_S(\theta)$, i.e.

$$\begin{cases}\dfrac{\mathrm{d}\theta}{\mathrm{d}t}=-\nabla_\theta R_S(\theta),\\ \theta(0)=\theta_0.\end{cases}$$

Back propagation. For every $l\in[L]$, we define the error vectors $z_\theta^{[l]}=\nabla_{f^{[l]}}\ell$ and the feature gradients $g_\theta^{[L]}=\mathbf{1}$ along with $g_\theta^{[l]}=\sigma^{(1)}(W^{[l]}f_\theta^{[l-1]}+b^{[l]})$ for $l\in[L-1]$, where $\sigma^{(1)}$ signifies the first derivative of σ. Furthermore, $f_\theta^{[l]}$, for $l\in[L]$, are referred to as feature vectors. We denote the collections of feature vectors, feature gradients, and error vectors

by $\boldsymbol{F}_{\boldsymbol{\theta}}=\{\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}\}_{l=1}^{L}$, $\boldsymbol{G}_{\boldsymbol{\theta}}=\{\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]}\}_{l=1}^{L}$, $\boldsymbol{Z}_{\boldsymbol{\theta}}=\{\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]}\}_{l=1}^{L}$, respectively. The gradients can be computed employing backpropagation as follows:

$$\begin{cases} \boldsymbol{z}_{\boldsymbol{\theta}}^{[L]}=\nabla\ell, \\ \boldsymbol{z}_{\boldsymbol{\theta}}^{[l]}=\left(\boldsymbol{W}^{[l+1]}\right)^{\top}\left(\boldsymbol{z}_{\boldsymbol{\theta}}^{[l+1]}\circ\boldsymbol{g}_{\boldsymbol{\theta}}^{[l+1]}\right), & l\in[L-1], \\ \nabla_{\boldsymbol{W}^{[l]}}\ell=\left(\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]}\circ\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]}\right)\left(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]}\right)^{\top}, & l\in[L], \\ \nabla_{\boldsymbol{b}^{[l]}}\ell=\boldsymbol{z}_{\boldsymbol{\theta}}^{[l]}\circ\boldsymbol{g}_{\boldsymbol{\theta}}^{[l]}, & l\in[L]. \end{cases} \tag{3.1}$$

4 Theory of embedding principle in depth

Consider a neural network $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{\theta}$ represents the set of all network parameters and $\boldsymbol{x}\in\mathbb{R}^d$ is the input. We summarize the assumptions for all our theoretical results in this work as follows.

Assumption 4.1. (i) L-layer ($L\geq 1$) fully-connected NN.

(ii) Training data $S=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^{n}$ for $n\in\mathbb{Z}^+$.

(iii) Empirical risk $R_S(\boldsymbol{\theta})=\mathbb{E}_S\ell(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}),\boldsymbol{y})$.

(iv) Activation function σ has a non-constant linear segment, e.g. ReLU, leaky-ReLU and ELU.

(v) Loss function ℓ and activation function σ are subdifferentiable, i.e. a unique subgradient can be assigned to each non-differentiable point.

Remark 4.1. For general smooth activations without a linear segment, e.g. tanh, our results hold in the sense of approximation because they are arbitrarily close to linear in a sufficiently small interval (for instance, around 0). Therefore, we also demonstrate our results using the tanh activation in the subsequent numerical experiments.

Definition 4.1 (Affine Subdomain)**.** *For an activation σ with a non-constant linear segment, an affine subdomain of σ is an open interval (a,b) satisfying that there exist $\lambda,\mu\in\mathbb{R}$ ($\lambda\neq 0$), $\sigma(x)=\lambda x+\mu$ for any $x\in(a,b)$.*

4.1 Lifting operator

We begin by introducing a lifting operator, as illustrated in Fig. 2.

Definition 4.2 (One-Layer Lifting)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* NN$'(\{m_l'\},l\in\{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\})$. *The one-layer lifting, denoted as $\mathcal{T}_S$, is a function that transforms any parameter $\boldsymbol{\theta}=(\boldsymbol{W}^{[1]},\boldsymbol{b}^{[1]},\cdots,\boldsymbol{W}^{[L]},\boldsymbol{b}^{[L]})$ of* NN *into a set $\mathcal{M}$ within the parameter space of* NN$'$. *Formally, $\mathcal{M}$ (where $\mathcal{M}:=\mathcal{T}_S(\boldsymbol{\theta})$) represents a collection of all possible parameters $\boldsymbol{\theta}'$ of* NN$'$ *that satisfying the following three conditions:*

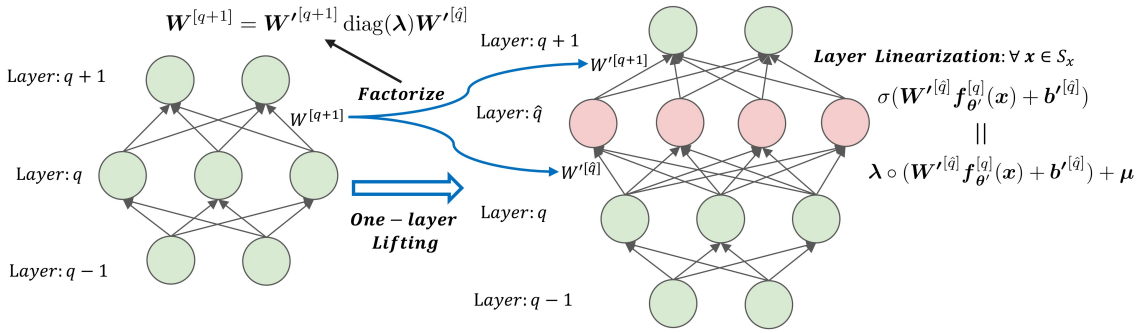


Figure 2: Illustration of one-layer lifting. The pink layer is inserted into the left network to get the right network. The input parameters $W'^{[\hat{q}]}$ and output parameters $W'^{[q+1]}$ of the inserted layer are obtained by factorizing the input parameters $W^{[q+1]}$ of $(q+1)$-th layer in the left network to satisfy layer linearization and output preserving conditions.

(i) Local-in-layer condition: Weights of each layer in NN' *are inherited from* NN *except for layer* $\hat{q}$ *and* $q+1$*, i.e.*

$$\begin{cases} \theta'|_l = \theta|_l, \quad l \in [q] \cup [q+2:L], \\ \theta'|_{\hat{q}} = \left(W'^{[\hat{q}]}, b'^{[\hat{q}]}\right) \in \mathbb{R}^{m'_{\hat{q}} \times m'_{q-1}} \times \mathbb{R}^{m'_{\hat{q}}}, \\ \theta'|_{q+1} = \left(W'^{[q+1]}, b'^{[q+1]}\right) \in \mathbb{R}^{m'_{q+1} \times m'_{\hat{q}}} \times \mathbb{R}^{m'_{q+1}}. \end{cases} \tag{4.1}$$

(ii) Layer linearization condition: For any $j \in [m_{\hat{q}}]$*, there exists an affine subdomain* (a_j, b_j) *of* σ *associated with* λ_j, μ_j *such that the j-th component* $(W'^{[\hat{q}]} f_{\theta'}^{[q]}(x) + b'^{[\hat{q}]})_j \in (a_j, b_j)$ *for any* $x \in S_x$*.*

(iii) Output preserving condition

$$\begin{cases} W'^{[q+1]}\,\mathrm{diag}(\lambda)W'^{[\hat{q}]} = W^{[q+1]}, \\ W'^{[q+1]}\,\mathrm{diag}(\lambda)b'^{[\hat{q}]} + W'^{[q+1]}\mu + b'^{[q+1]} = b^{[q+1]}, \end{cases} \tag{4.2}$$

where

$$\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_{m_{\hat{q}}}]^\top \in \mathbb{R}^{m'_{\hat{q}}}, \quad \mu = [\mu_1, \mu_2, \cdots, \mu_{m_{\hat{q}}}]^\top \in \mathbb{R}^{m'_{\hat{q}}},$$

and $\mathrm{diag}(\lambda)$ *denotes the diagonal matrix formed by vector* λ*.*

Remark 4.2. Intuitively, the lifting operator described above elevates a point θ from NN to a set $\mathcal{M}$ within a higher-dimensional space. In general, $\mathcal{M}$ is a finite union of manifolds instead of a set of isolated points. To highlight this point, we refer to the mapped set $\mathcal{M} := \mathcal{T}_S(\theta)$ as a "manifold" with slight abuse of terminology throughout this paper. For a similar reason, we also refer to the set of critical points sharing the same loss value as "critical manifold".

Remark 4.3. The one-layer lifting operator describes the generic process of mapping a point from a lower-dimensional space to a set within a higher-dimensional space. Any $\boldsymbol{\theta}'$ that belongs to the mapped set as outlined in Definition 4.2 is termed a "lifted point". For clarity, when discussing specific operations that map a point $\boldsymbol{\theta}$ of NN to a specific point $\boldsymbol{\theta}' \in \mathcal{T}_S(\boldsymbol{\theta})$ of NN$'$, we use the term "embedding" to describe this operation.

As illustrated in Fig. 2, a one-layer lifting is realized by inserting a hidden layer, depicted here as the pink layer ($\hat{q}$-th layer) in the right network. The outcome of a one-layer lifting is a manifold of the parameter space of the right network, which comprises each parameter vector that satisfies the following constraints: The parameters $\boldsymbol{W}'^{[\hat{q}]}, \boldsymbol{b}'^{[\hat{q}]}$ of the inserted layer meet the layer linearization condition, ensuring this layer operates like a linear layer when applied to the training inputs $S_{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i=1}^n$. Additionally, the parameters $\boldsymbol{W}'^{[q+1]}, \boldsymbol{b}'^{[q+1]}$ of $(q+1)$-th layer fulfill the output preserving condition, making the composition of the $\hat{q}$-th layer and the $(q+1)$-th layer in the right network equivalent to the $(q+1)$-th layer in the left network.

Because the factorized weights satisfying both layer linearization and output preserving conditions always exist, we have the following existence result for one-layer lifting.

Lemma 4.1 (Existence of One-Layer Lifting)**.** *Given data S, an* NN$(\{m_l\}_{l=0}^L)$ *and its one-layer deeper counterpart,* NN$'(\{m_l'\}, l \in \{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\})$*, the one-layer lifting $\mathcal{T}_S$ exists, i.e. $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$ is not empty for any parameter $\boldsymbol{\theta}_{\text{shal}}$ of* NN*.*

The proof of this lemma is given in Appendix A (Lemma A.1).

4.2 Embedding principle in depth

The multi-layer lifting is defined as the composition of multiple one-layer liftings. Consequently, the parameters of any neural network can be lifted to a parameter manifold of any deeper neural network through a multi-layer lifting. Both one-layer and multi-layer liftings exhibit two key attributes: network properties preservation and criticality preservation. To demonstrate these two properties, we first consider the following lemma.

Lemma 4.2 (Computation of Feature Vectors, Feature Gradients and Error Vectors)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^L)$ *and its one-layer deeper counterpart,* NN$'\big(\{m_l'\}, l \in \{0,1,2,\ldots, q,\hat{q},q+1,\ldots,L\}\big)$*. Let $\mathcal{T}_S$ denote the one-layer lifting and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. Then, for any lifted point $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$, the following conditions hold: There exist $\boldsymbol{\lambda}, \boldsymbol{\mu} \in \mathbb{R}^{m'_{\hat{q}}}$ such that for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$,*

(i) feature vectors in $\boldsymbol{F}_{\boldsymbol{\theta}'_{\text{deep}}}$

$$
\begin{aligned}
&\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}), \quad l \in [L], \\
&\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \operatorname{diag}(\boldsymbol{\lambda}) \left(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]} \right) + \boldsymbol{\mu},
\end{aligned}
$$

(ii) feature gradients in $\boldsymbol{G}_{\boldsymbol{\theta}'_{\text{deep}}}$

$$\begin{aligned}\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \boldsymbol{g}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}), \quad l \in [L],\\ \boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \boldsymbol{\lambda},\end{aligned}$$

(iii) error vectors in $\boldsymbol{Z}_{\boldsymbol{\theta}'_{\text{deep}}}$

$$\begin{aligned}\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \boldsymbol{z}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}), \quad l \in [q-1] \cup [q+1:L],\\ \boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \left(\boldsymbol{W}'^{[q+1]}\right)^{\top}\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right),\\ \boldsymbol{z}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \left(\boldsymbol{W}'^{[\hat{q}]}\right)^{\top}\left(\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{\lambda}\right).\end{aligned}$$

The proof of this lemma is given in Appendix A (Lemma A.2).

Using Lemma 4.2, we can promptly derive the property of network preservation. This underlines the preservation of the output function by the lifting process during the transformation from a shallower to a deeper neural network.

Proposition 4.1 (Network Properties Preserving)**.** *Given data* S*, consider an* $\text{NN}(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* $\text{NN}'(\{m'_l\}, l \in \{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\})$*. Let* $\mathcal{T}_S$ *denote the one-layer lifting and* $\boldsymbol{\theta}_{\text{shal}}$ *be any parameter of* NN*. Then, for any lifted point* $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$*, the following conditions hold:*

(i) outputs are preserved: $f_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})$ *for* $\boldsymbol{x} \in S_{\boldsymbol{x}}$*,*

(ii) empirical risk is preserved: $R_S(\boldsymbol{\theta}'_{\text{deep}}) = R_S(\boldsymbol{\theta}_{\text{shal}})$*,*

(iii) network representations are preserved for all layers,

$$\text{span}\left\{\left\{\left(\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m'_{\hat{q}}]} \cup \{\boldsymbol{1}\}\right\} = \text{span}\left\{\left\{\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m_q]} \cup \{\boldsymbol{1}\}\right\},$$

and for the other index $l \in [L]$*,*

$$\text{span}\left\{\left\{\left(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m'_l]} \cup \{\boldsymbol{1}\}\right\} = \text{span}\left\{\left\{\left(\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m_l]} \cup \{\boldsymbol{1}\}\right\},$$

where

$$\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{X}) = \left[\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x}_1), \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x}_2), \cdots, \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\right]^{\top} \in \mathbb{R}^{n\times m'_{\hat{q}}},$$

and $\boldsymbol{1} \in \mathbb{R}^n$ *is the all-ones vector.*

The proof of this proposition is given in Appendix A (Proposition A.1).

The most significant characteristic of the lifting operator is its preservation of criticality. That is to say, if a shallow network is at a critical point, it will still be at a critical point when transformed into a deeper network through the lifting operator.

Proposition 4.2 (Criticality Preserving)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* NN$'\big(\{m_l'\}, l\in\{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\}\big)$*. Let $\mathcal{T}_S$ denote the one-layer lifting and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. If $\boldsymbol{\theta}_{\text{shal}}$ of* NN *satisfies $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}_{\text{shal}})=\mathbf{0}$, then $\nabla_{\boldsymbol{\theta}'} R_S(\boldsymbol{\theta}'_{\text{deep}})=\mathbf{0}$ for any lifted point $\boldsymbol{\theta}'_{\text{deep}}\in\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$.*

The proof of this proposition is given in Appendix A (Proposition A.2).

Owing to the criticality preserving property, we term one-layer or multi-layer lifting as critical lifting. With the aforementioned results, we now establish an embedding principle in depth, which can be intuitively described as follows: The loss landscape of any DNN contains a hierarchy of critical manifolds, each of which is lifted from the critical points of the loss landscapes of all its shallower counterparts.

Theorem 4.1 (Embedding Principle in Depth)**.** *Given data S and an* NN$'(\{m_l'\}_{l=0}^{L'})$*, for any parameter $\boldsymbol{\theta}_{\text{c}}$ of any shallower* NN$(\{m_l\}_{l=0}^{L})$ *satisfying $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}_{\text{c}})=\mathbf{0}$, there exists parameter $\boldsymbol{\theta}'_{\text{c}}$ in the loss landscape of* NN$'(\{m_l'\}_{l=0}^{L'})$ *satisfying the following conditions:*

(i) $\boldsymbol{f}_{\boldsymbol{\theta}'_{\text{c}}}(\boldsymbol{x})=\boldsymbol{f}_{\boldsymbol{\theta}_{\text{c}}}(\boldsymbol{x})$ *for* $\boldsymbol{x}\in S_{\boldsymbol{x}}$*,*

(ii) $\nabla_{\boldsymbol{\theta}'} R_S(\boldsymbol{\theta}'_{\text{c}})=\mathbf{0}$*.*

The proof of this theorem is given in Appendix A (Theorem A.1).

Remark 4.4. Although we only prove output preserving for training inputs, it is important to note that the output function of the neural network is indeed preserved over a broader area of the input space including at least a neighbourhood of each training input (see Proposition A.4 in Appendix A for proof). Consequently, if the training dataset is sufficiently large and representative, then the lifting operator effectively preserves the generalization performance.

Leveraging critical lifting, the aforementioned embedding principle in depth provides a clear picture about the hierarchical structure of critical points/manifolds in depth within a DNN's loss landscape. This hierarchical structure profoundly influences the nonlinear training behavior of a deep network, as any nearby training trajectory tends to gravitate towards these points/manifolds.

Critical lifting delineates the relationship between the critical points of deep networks and their shallow counterparts. Notably, it also preserves the positive and negative inertia indices of the Hessian matrix.

Proposition 4.3 (Positive and Negative Index of Inertia Preserving)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^{L})$*, and its deeper counterpart* NN$'(\{m_l'\}_{l=0}^{L'})$*. Let $\mathcal{T}_S$ denote the corresponding critical lifting and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. Then, for any critical embedding $\mathcal{E}:\mathbb{R}^M\to\mathbb{R}^{M'}$*

resulting from $\mathcal{T}_S$, denote $\boldsymbol{\theta}'_{\text{deep}} := \mathcal{E}(\boldsymbol{\theta}_{\text{shal}})$, the number of positive and negative eigenvalues of Hessian matrix $\boldsymbol{H}_S(\boldsymbol{\theta}'_{\text{deep}})$ equals the counterparts of $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}})$.

The proof of this proposition is given in Appendix A (Proposition A.5).

Utilizing Proposition 4.3, we immediately conclude that through critical lifting, the degeneracy of a critical point will increase.

Corollary 4.1 (Incremental Degeneracy of Critical Point Through Lifting)**.** *Given data S, consider an* $\text{NN}(\{m_l\}_{l=0}^{L})$*, and its deeper counterpart* $\text{NN}'(\{m'_l\}_{l=0}^{L'})$*. Let $\mathcal{T}_S$ denote the corresponding critical lifting and $\boldsymbol{\theta}_{\text{shal}} \in \mathbb{R}^M$ be a critical point of* NN*. Then, any lifted point $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}}) \subseteq \mathbb{R}^{M'}$ possesses $M'-M$ additional degrees of degeneracy in comparison to $\boldsymbol{\theta}_{\text{shal}}$.*

The proof of this corollary is given in Appendix A (Corollary A.1).

It is important to note that critical lifting is data-dependent, and the nature of this data-dependence is characterized by the following proposition.

Proposition 4.4 (Data and Critical Lifting)**.** *Given data S, S', consider an* $\text{NN}(\{m_l\}_{l=0}^{L})$ *and its deeper counterpart* $\text{NN}'(\{m'_l\}_{l=0}^{L'}$*. Let $\mathcal{T}_S$ and $\mathcal{T}_{S'}$ denote the respective critical liftings and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. If $S' \subseteq S$, then $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}}) \subseteq \mathcal{T}_{S'}(\boldsymbol{\theta}_{\text{shal}})$.*

The proof of this proposition is given in Appendix A (Proposition A.3).

This result indicates that increasing training data shrinks any lifted manifold to its subset. The implication is that enlarging the training dataset is a viable strategy for diminishing the lifted critical manifolds, which can consequently expedite the decay of the training loss, as demonstrated in the subsequent experimental study.

5 Numerical experiments

The theory of the embedding principle in depth highlights the existence of a class of "simple" critical points inherited from shallower neural networks. A natural question arises as to whether deep networks encounter these lifted critical points. Moreover, it is essential to understand the influence of these critical points on the training dynamics. In this section, we conduct comprehensive experiments to study these questions. In Section 5.1, we briefly describe the experimental setup. Section 5.2 is dedicated to a detailed comparison of the training dynamics between deep and shallow networks. Section 5.3 investigates the impact of batch normalization and larger datasets on optimization. Lastly, in Section 5.4, we explore the practical application of layer-wise network pruning.

5.1 Experimental setup

Measuring layer linearization by minimal Pearson correlation (MPC). To detect lifted critical points in our experiments, we propose a method to measure their key feature,

i.e. layer linearization. Let $\tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]} + \boldsymbol{b}^{[l]} \in \mathbb{R}^{m_l}$ and $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]} = \sigma(\tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]}) \in \mathbb{R}^{m_l}$ denote the input and output of neurons in layer l, respectively. For each neuron in a layer, the absolute value of the Pearson correlation coefficient is utilized to measure the extent of linearization for each neuron. By taking the minimum over the whole layer, we obtain the following measure of the extent of linearization for the l-th layer:

$$\mathrm{MPC}\left(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}, \tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]}\right) = \min_{j \in [m_l]} \left| \rho\left(\left(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}\right)_j, \left(\tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]}\right)_j \right) \right| \in [0,1], \tag{5.1}$$

where $(\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]})_j, (\tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]})_j$ represent the j-th components of $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}$ and $\tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]}$, respectively, and $\rho((\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]})_j, (\tilde{\boldsymbol{f}}_{\boldsymbol{\theta}}^{[l]})_j)$ denotes the Pearson correlation coefficient.

Remark 5.1. To determine whether a critical point in an experiment is a lifted critical point, two conditions must be met: (i) the MPC of a layer equals 1, and (ii) merging that layer with the subsequent layer leads to a critical point of the shallower neural network.

In the subsequent experiments conducted, fully-connected networks are predominantly utilized. The input dimension, denoted as d, and output dimension, represented by d', are determined by the respective training dataset. Each hidden layer has the same width m. All parameters are initialized by a Gaussian distribution with mean zero and variance specified in each experiment. To investigate the training behavior of DNNs with feature learning, we employ relatively small initializations to enhance the nonlinearity of training, which stays away from the neural tangent kernel (NTK) regime. To meticulously examine the dynamics during the training process, a full-batch gradient descent approach is employed, combined with a small learning rate. More details of experiments are presented in Appendix C.

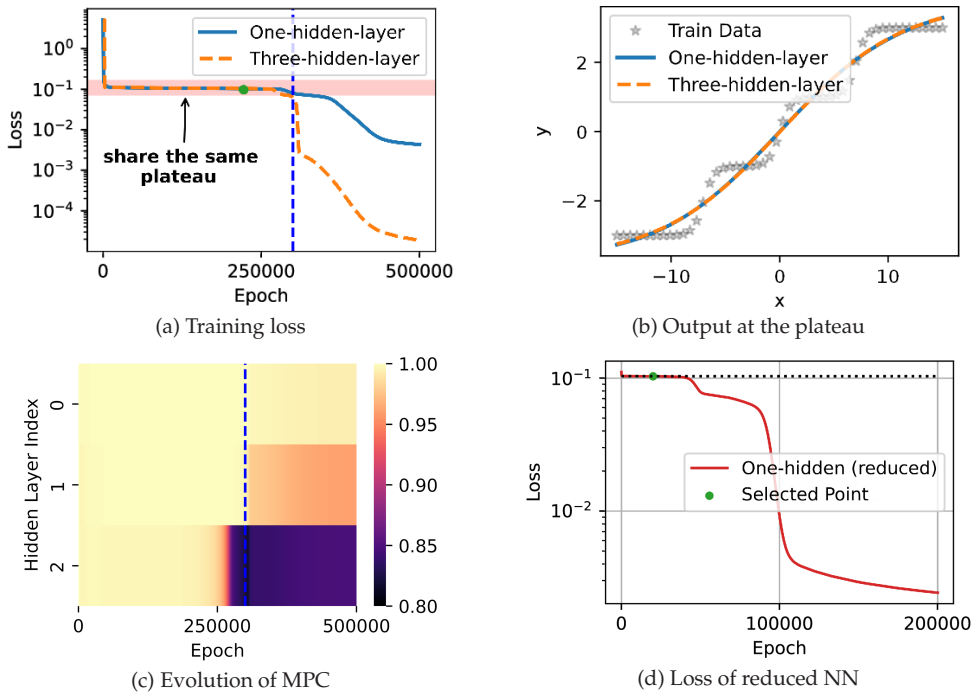
5.2 Training dynamics of deep and shallow neural networks

5.2.1 Deep neural networks encounter lifted critical points during practical training

To investigate whether deep neural networks encounter lifted critical points during training, we train tanh NNs with different depths (width $m=50$) on the data shown in Fig. 3(b) and the Iris dataset in Fig. 4. During training, we trace the evolution of the MPC for each hidden layer computed using Eq. (5.1).

As depicted in Fig. 3(a), the three-hidden-layer NN first stagnates at the same loss value as the single-hidden-layer NN, displaying nearly the same output function as illustrated in Fig. 3(b). According to Fig. 3(c), the first two hidden layers exhibit strong linearity during stagnation. In Fig. 3(d), we observe that merging the effectively linear layers using Eq. (4.2) results in a critical point of the single-hidden-layer NN.

A similar phenomenon is observed for the Iris dataset in Fig. 4. As shown in Fig. 4(a), the three-hidden-layer NN first stagnates at the same loss value as the single-hidden-layer NN, displaying nearly the same training and test accuracy as illustrated in Fig. 4(b).



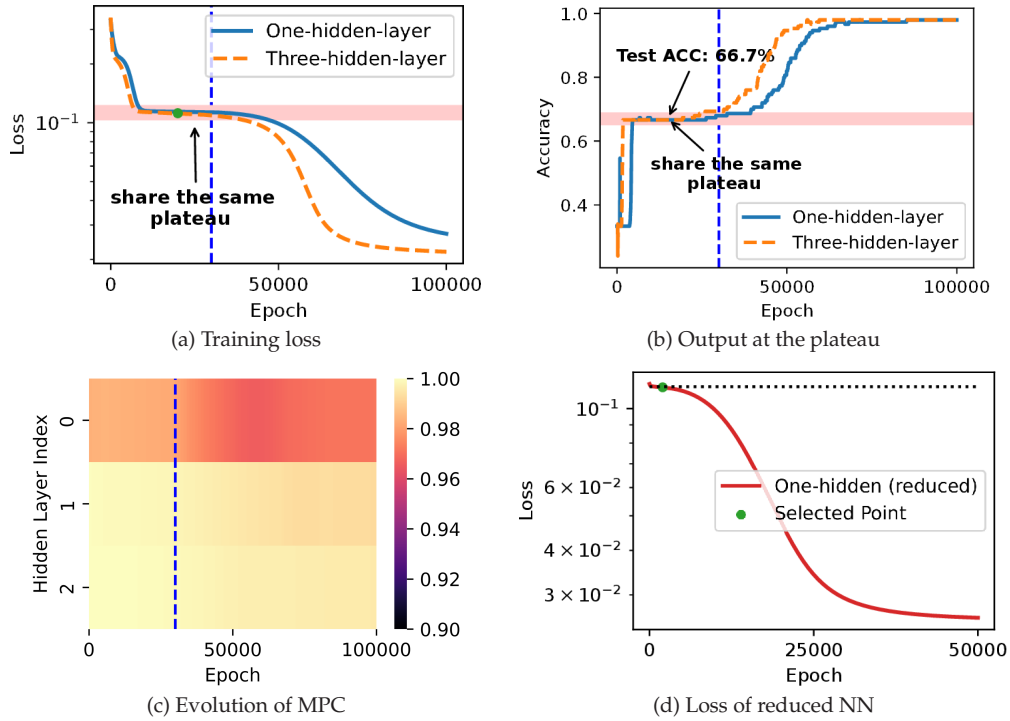
(a) Training loss (b) Output at the plateau

(c) Evolution of MPC (d) Loss of reduced NN

Figure 3: Deep neural networks encounter lifted critical points during training on synthetic data. (a) The training loss for single-hidden-layer and three-hidden-layer NNs with width $m=50$. (b) The outputs of NNs with different depths at the same loss value indicated by the colored span in (a). (c) The extent of layer linearization for different hidden layers during the training process of the three-hidden-layer NN. (d) Training loss trajectory of the reduced single-hidden- layer NN. The green dot in (a) and (c) is selected as a representative for comparison.

Upon examining the confusion matrices of the networks at this plateau, we observe that both networks correctly classify two of the three classes and completely misclassify the third class, achieving an accuracy of 66.7%. According to Fig. 4(c), the last two hidden layers exhibit strong linearity during stagnation. In Fig. 4(d), we find that merging the effectively linear layers using Eq. (4.2) results in a critical point of the single-hidden-layer NN.

We observe similar phenomena for ReLU NNs and residual-connected NNs (ResNets) in Figs. 10 and 11 in Appendix B. These results confirm that deep NNs indeed encounter critical points lifted from shallower NNs with small initialization. Moreover, the study conducted by [15] utilized linear centered kernel alignment (CKA) as a metric to assess the similarity between different layers. Their experiments on CIFAR-10 and ImageNet-1000 revealed that a "block structure" emerges when the network size is much larger than the dataset size, with many layers exhibiting a high degree of similarity. In Proposition A.6 of Appendix A, we prove that this similarity between layers indeed reflects the degree of linear correlation between representations across layers. We further investigate the impact of initialization and dataset size on layer linearization in Appendix B (see Fig. 12). Generally speaking, it is common to observe layer linearization when a deep



(a) Training loss (b) Output at the plateau

(c) Evolution of MPC (d) Loss of reduced NN

Figure 4: Deep neural networks encounter lifted critical points during training on Iris data. (a) The training loss for single-hidden-layer and three-hidden-layer NNs with width $m=50$. (b) The training accuracy of NNs with different depths at the same loss value indicated by the colored span in (a). The accuracy plateau is at 66.7% for both train and test sets. (c) The extent of layer linearization for different hidden layers during the training process of the three-hidden-layer NN. (d) Training loss trajectory of the reduced single-hidden-layer NN. The green dot in (a) and (c) is selected as a representative for comparison.

network is trained on a simple task with small initialization. This occurrence of layer linearization significantly reduces the network's complexity and thus may be crucial in contributing to the generalization of deep networks.

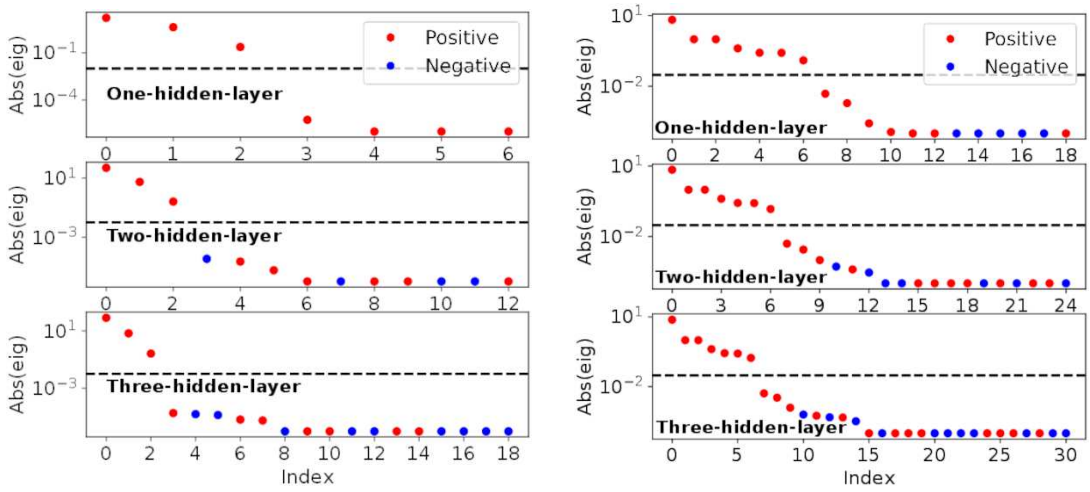
Remark 5.2. Generally, reducing the scale of initialization enhances the nonlinear feature learning dynamics of DNNs. With a properly small initialization, the training dynamics tend to experience lifted critical points from shallower NNs, which implicitly help control the complexity of DNNs during training. However, a too small initialization scale may result in undesired prolonged stagnation at critical points. Remarkably, standard initialization schemes like Xavier/Kaiming-He strike a balance between feature learning and training efficiency with a proper initialization scale, ensuring that the training dynamics are both efficient and sufficiently nonlinear.

5.2.2 Incremental degeneracy of critical points through embedding

Empirical studies have shown that the Hessian matrix of the minimizer, derived from training, possesses a significant count of zero or near-zero eigenvalues, highlighting the

existence of highly degenerate critical points within the loss landscape [19]. Our findings, as illustrated in Figs. 3 and 4, imply that deep networks often encounter critical points inherited from their shallower counterparts. To empirically validate that lifted critical points have higher degeneracy, we design experiments to compute the eigenvalues of their respective Hessian matrices at the empirical critical points.

Specifically, we train a single-hidden-layer ReLU NN with width $m=2$ to learn the data in Fig. 3(b) shown in Fig. 5(a) or the Iris dataset in Fig. 5(b) to a empirical critical point (the L_1 norm of the gradient $\leq 10^{-4}$). We then embed this critical point through a one-layer embedding and a two-layer embedding to NNs with 2 hidden layers and 3 hidden layers, with each hidden layer having the same width, respectively. To compute the eigenvalues of the Hessian matrix with a large condition number accurately, we conducted 100 random orthogonal similarity transformations on the matrix. We took the average from these 100 trials to obtain a more reliable set of eigenvalues. We then pinpoint locations where there are evident gaps in eigenvalue magnitudes, as delineated by the auxiliary line in Fig. 5. This allows for differentiation between zero and non-zero eigenvalues, serving as a mechanism to ascertain empirical degeneracy. Detailed methodology can be found in Appendix C. As illustrated in Fig. 5, each embedding step introduces six more zero eigenvalues to the Hessian matrix due to the introduction of two neurons, resulting in six additional parameters. This finding is consistent with Proposition 4.3 and potentially elucidates the origin of a particular type of degeneracy at critical points within the loss landscape.



(a) Synthetic data (ReLU) (b) Iris data (ReLU)

Figure 5: Incremental degeneracy of critical points through embedding. (a,b) The eigenvalues of Hessian of ReLU NNs at the critical points embedded from the single hidden layer NN for learning data in Fig. 3(b) and Iris data, respectively. The results for each plot are averaged over 100 random orthogonal similarity transformations. The auxiliary dashed lines in (a,b) delineate the empirical boundary between zero and non-zero eigenvalues. We perform the embedding operation by factorizing one hidden layer into k hidden layers ($k=2,3$), whose input weights are identity and biases are selected to translate the input range into the affine subdomain.

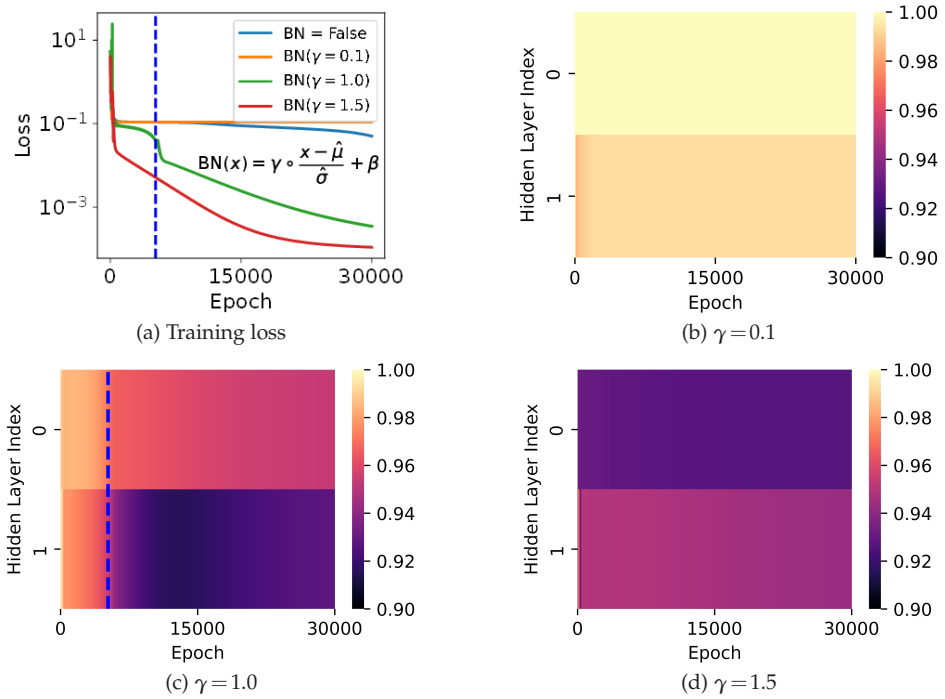
5.3 The effects of batch normalization and larger dataset

Since layer linearization is a key feature in encountering lifted critical points and results in slower training, we investigate how batch normalization and larger training sets can affect layer linearization and subsequently lead to accelerated training.

5.3.1 Batch normalization avoids lifted critical points

Batch normalization (BN) normalizes the layers' inputs by re-centering and re-scaling: $\mathrm{BN}(\boldsymbol{x}) = \boldsymbol{\gamma} \circ ((\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}})/\hat{\boldsymbol{\sigma}}_{\mathcal{B}}) + \boldsymbol{\beta}$ with a default initialization $\boldsymbol{\gamma} = \mathbf{1}, \boldsymbol{\beta} = \mathbf{0}$. Empirically, using BN can greatly speed up NN training. Intuitively, when a neuron's input range becomes too small, its nonlinear activation function behaves effectively like a linear function. In such cases, batch normalization can enhance the nonlinearity of each neuron by effectively rescaling its input range to $\mathcal{O}(1)$ and thus suppress layer linearization. As embedding principle in depth unravels a large family of lifted critical points with layer linearization, avoiding these critical points through suppressing layer linearization may be an important mechanism underlying the training efficiency of BN in practice.

To verify this mechanism, we perform the following experiment shown in Fig. 6. We train a 2-hidden-layer tanh NN (width $m = 50$) to learn the data in Fig. 3(b), and com-



(a) Training loss (b) $\gamma = 0.1$ (c) $\gamma = 1.0$ (d) $\gamma = 1.5$

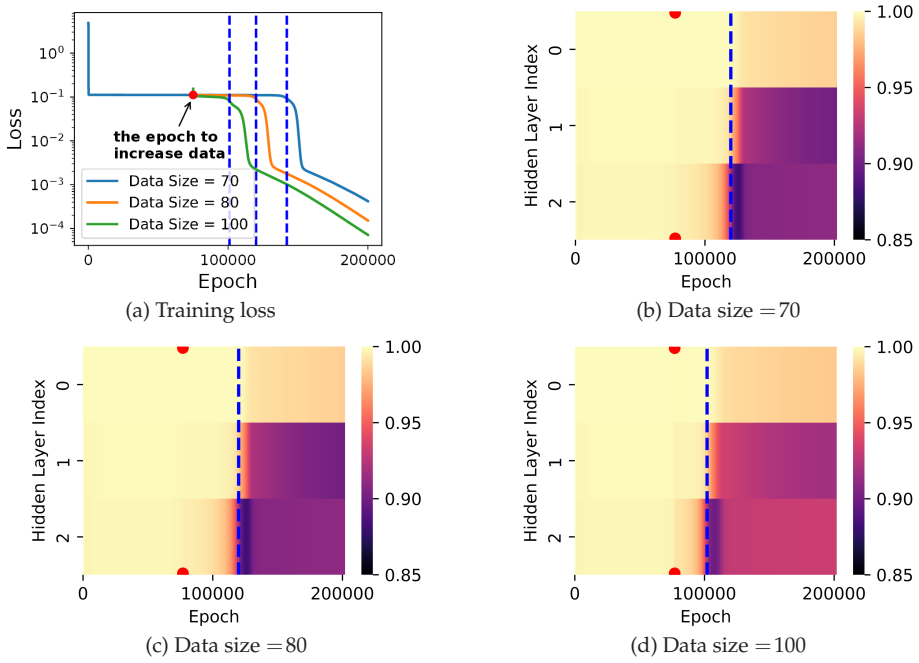
Figure 6: Batch normalization avoids lifted critical points during training. (a) Trajectories of training loss without BN and with BN of different initial values. (b-d) The extent of layer linearization for all hidden layers with BN of scaling parameter γ initialized at $0.1, 1.0$ or 1.5, respectively. The auxiliary dash lines in (a) and (c) correspond to the same epoch.

pare the training trajectories without BN and with BN of scaling parameter γ initialized at 0.1, 1.0, or 1.5. Conforming with our intuition, a larger γ better suppresses layer linearization throughout the training as shown in Figs. 6(b)-6(d). Moreover, the stagnation is significantly alleviated with a larger γ. There is even no stagnation for γ initialized at 1.5, signifying complete avoidance of lifted critical points as predicted by above mechanism.

5.3.2 Optimization benefit of larger dataset

Proposition 4.4 characterizes the data dependency of critical lifting. The intuition is that a larger dataset increases the difficulty of layer linearization, thus helping reduce the critical manifolds. This result provides a seemingly counter-intuitive prediction that larger dataset may be more easily fitted due to the reduced critical manifolds lifted from shallower NNs. This prediction is verified by the following experiment in Fig. 7.

We train a tanh NN with 3 hidden layers (width $m=50$) to learn data (data size $n=70$) of Fig. 3(b) to a critical point (the red point in Fig. 7(a)). We then continue (blue curve) or switch to larger datasets (orange and green curves) for training. As shown in Fig. 7(a), more training data leads to faster escape from the lifted critical point. Figs. 7(b)-7(d) further trace the extent of layer linearization for each hidden layer on datasets of different



(a) Training loss (b) Data size $=70$

(c) Data size $=80$ (d) Data size $=100$

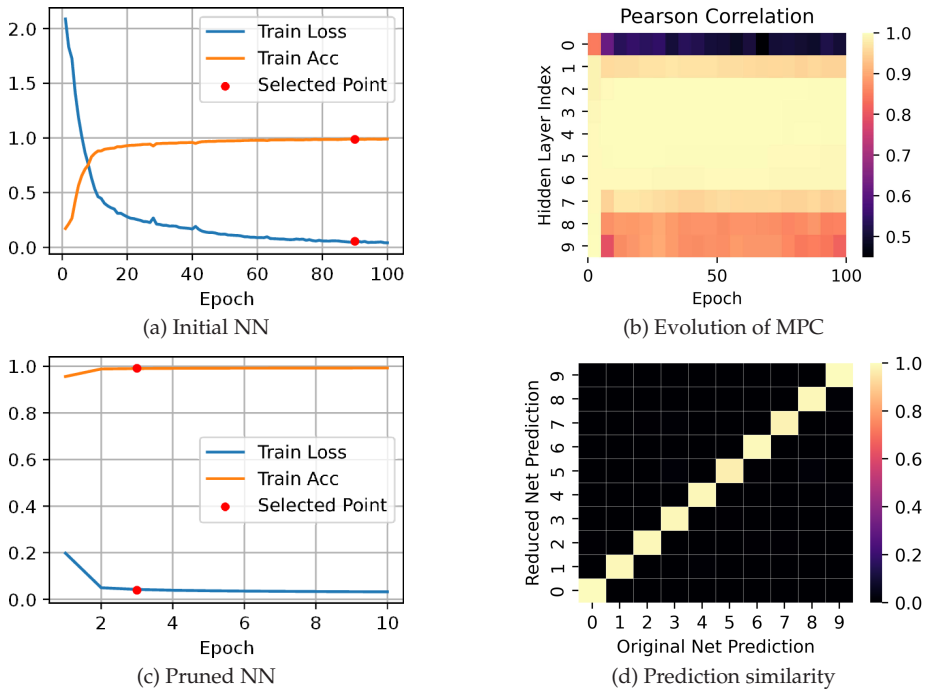
Figure 7: Optimization benefit of larger dataset. (a) The training loss of three-hidden-layer NN with width $m=50$ for learning data of Fig. 3(b). We sample 70, 80 and 100 data points equally spaced around 0, respectively. The red dot in (a) is selected for switching dataset. (b-d) The extent of layer linearization for all hidden layers on datasets of different size, respectively. The red dots correspond to the epoch to switch dataset. The auxiliary dash lines correspond to the epoch where NNs escape from the lifted critical manifold.

sizes, respectively. From Figs. 7(c)-7(d), we can clearly see that more data facilitate the expression of nonlinearity of hidden layers (see the abrupt reductions of MPC at the red dot). This helps the network to escape the critical manifold lifted from a single-hidden-layer NN, which aligns with the implications of Proposition 4.4.

5.4 Network pruning

5.4.1 Layer pruning of DNNs with layer linearization

The embedding principle in depth predicts a family of critical points with layer linearization. These critical points intrinsically come from shallower NNs, thus possessing good layer pruning potential. To realize such pruning potential in practice, we propose the method of detecting and merging effectively linear layers, which works as follows. We train a deep 10-hidden-layer tanh NN (width $m = 50$) on the MNIST dataset. At the red dot in Fig. 8(a), the training loss decreases very slowly, presumably is very close to a global minimum. As shown in Fig. 8(b), there are 5 effective linear layers (MPC$>$0.99) at this point. We merge these effective linear layers by properly multiplying their weights, thereby pruning the NN of 10 hidden layers to 5 hidden layers. The parameters before



(a) Initial NN (b) Evolution of MPC

(c) Pruned NN (d) Prediction similarity

Figure 8: Layer pruning of DNNs with layer linearization. (a) The training process of the original 10-hidden-layer network on MNIST dataset. The red dot is selected for layer pruning. (b) The extent of layer linearization for all hidden layers during the training process. (c) The training process after layer pruning. (d) Prediction similarity between initial and reduced network on the test dataset. For each grid, color indicates the ratio of that prediction pair (i,j) over all samples predicted as j by the original 10-hidden-layer NN.

reduction is denoted by $\boldsymbol{\theta}_{\text{ori}}$ and after reduction by $\boldsymbol{\theta}_{\text{redu}}$. We further train the pruned NN from $\boldsymbol{\theta}_{\text{redu}}$ as shown in Fig. 8(c) which quickly fall into the same loss value as the red point in Fig. 8(a). We then compare the prediction between original model and the pruned model at the corresponding red point on 10000 test data as shown in Fig. 8(d). Although our critical lifting does not preserve the output function over the entire domain of input, we still observe well agreement of these two models (overall $\sim$98.54%), which implies that this reduction can approximately preserve the generalization performance (95.4% to 95.27%).

6 Discussion

6.1 Differences between embedding principle in width and in depth

Our work draws inspiration from the embedding principle in width [32], but, for the first time, addresses the embedding relation in depth for DNNs. Though our depth-based critical lifting operator shares the same spirit with the width-based critical embedding operator in [32], it is important to note several key distinctions, as follows:

(i) **The target NN.** The depth-based critical lifting maps to the parameter space of a deeper NN whereas the width-based critical embedding maps to the parameter space of a wider NN.

(ii) **The requirement for the activation function.** The depth-based critical lifting requires a layer linearization condition, which can be satisfied for any activation with a non-constant linear segment, such as ReLU, leaky-ReLU, and ELU, and can be approximately satisfied for general smooth activations, including sigmoid, tanh, and gelu. On the other hand, the width-based critical embedding works for any activation function.

(iii) **The type of mapping.** The depth-based critical lifting is a set-valued function which maps any parameter vector to a manifold, whereas the width-based critical embedding is a vector-valued function.

(iv) **The output preserving property.** The depth-based critical lifting preserves the DNN outputs at the training dataset, whereas the width-based critical embedding preserves the DNN output function over the entire input domain.

(v) **The data dependency.** The depth-based critical lifting is data-dependent, whereas the width-based embedding operator is independent on data. For depth-based critical lifting, we prove that more data leads to reduced lifted manifolds in Proposition 4.4, whereas the width-based critical embedding is data-independent.

The key distinctions (iii)-(v) stem from the intrinsic differences in expressiveness when adding width versus depth to a DNN. Specifically, the function space of a narrow NN is strictly a subset of the function space of any wider NN, as referenced in [9,32]. However, a similar (embedding) relation regarding the expressiveness can generally only be expected in the sense of approximation when comparing shallower and deeper NNs.

6.2 Other network architectures

The embedding principle in depth is derived from the inherent layer stacking nature of deep neural networks, where each layer can be linearly factorized into multiple layers. Therefore, our theoretical findings proven for fully-connected DNNs can be naturally extended to other neural network architectures, such as convolutional neural networks and residual-connected neural networks. For instance, when dealing with a residual-connected network, the only alteration required to derive its one-layer lifting operator is a modification of the output-preserving condition (see Definition A.2 in Appendix A for details).

It should be noted that for residual-connected networks, simply assigning zero values to the parameters of the inserted block can create a trivially criticality-preserving lifting, which naturally satisfies the layer linearization condition. However, it is important to underscore that the practical experience of encountering these lifted critical points during training is a more crucial determinant of the importance of different lifting techniques than the mere existence of such embeddings. Indeed, such trivially lifted critical points are seldom observed in practice. On the contrary, the lifted critical points we propose, which are accompanied by layer linearization, are frequently observable in experimental setups, as illustrated in Fig. 11 in Appendix B. Therefore, the critical lifting operator proposed in this work is of special value for studying the practical training behavior of DNNs.

6.3 Simplicity bias

Our embedding principle in depth, combined with the prior embedding principle in width, explicitly characterizes the hierarchical structure of DNN loss landscape in both width and depth dimensions. This hierarchical structure highlights the potential for non-overfitting, even when a significantly large NN is employed to fit limited training data generated by a comparatively smaller (i.e. shallower and narrower) NN. The intuition here is that a large NN, guided by the hierarchy of "simple" critical points/manifolds lifted/embedded from shallower and narrower NNs, may learn a "simple" interpolation from a small NN through training. This potential is further corroborated by our numerical experiments shown in Figs. 3, 4, 6, 7 and 8, which indicate that the "effective" depth, i.e. the number of nonlinear layers, often gradually increases during the training of deep NNs. In light of these findings, it is tempting to conjecture that, with proper initialization, a large DNN could adaptively increase its "effective" depth and width based on the complexity of the training data. We will examine this conjecture in our future works.

7 Conclusion

In this paper, we discover an embedding principle in depth, establishing that the loss landscape of a deep NN inherits all critical points from shallower NNs. We introduce the

critical lifting operator that serves to prove this principle and provide comprehensive details about it. Furthermore, we offer empirical evidence demonstrating the vast insights provided by this principle, which contribute to the highly degenerate critical points of deep networks, the acceleration effect of batch normalization and larger datasets, and the process of layer pruning. It should be noted that the experiments conducted in this work serve as proofs of concept for these novel insights. Further systematic experimental studies are needed to gain a full understanding of the practical significance of these insights.

Overall, our discovery of the embedding principle in depth, together with the previous embedding principle in width, provides a comprehensive picture of the intrinsic hierarchical structure of the DNN loss landscape. This picture strongly supports the empirically observed similarities in training and generalization between NNs of varying sizes, thereby shedding light on the non-overfitting mystery of large NNs.

Appendix A. Proofs

In this section, we give all proofs for our theoretical results mentioned in the main text.

Definition A.1 (One-Layer Lifting)**.** *Given data S, consider an* $\mathrm{NN}(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* $\mathrm{NN}'(\{m_l'\}, l\in\{0,1,2,\dots,q,\hat{q},q+1,\dots,L\})$. *The one-layer lifting, denoted as $\mathcal{T}_S$, is a function that transforms any parameter $\boldsymbol{\theta}=(\boldsymbol{W}^{[1]},\boldsymbol{b}^{[1]},\cdots,\boldsymbol{W}^{[L]},\boldsymbol{b}^{[L]})$ of* NN *into a set $\mathcal{M}$ within the parameter space of* NN′. *Formally, $\mathcal{M}$ (where $\mathcal{M}:=\mathcal{T}_S(\boldsymbol{\theta})$) represents a collection of all possible parameters $\boldsymbol{\theta}'$ of* NN′ *that satisfying the following three conditions:*

(i) Local-in-layer condition: Weights of each layer in NN′ *are inherited from* NN *except for layer $\hat{q}$ and $q+1$, i.e.*

$$\begin{cases} \boldsymbol{\theta}'|_l=\boldsymbol{\theta}|_l \quad \text{for} \quad l\in[q]\cup[q+2:L],\\ \boldsymbol{\theta}'|_{\hat{q}}=(\boldsymbol{W}'^{[\hat{q}]},\boldsymbol{b}'^{[\hat{q}]})\in\mathbb{R}^{m'_{\hat{q}}\times m'_{q-1}}\times\mathbb{R}^{m'_{\hat{q}}},\\ \boldsymbol{\theta}'|_{q+1}=(\boldsymbol{W}'^{[q+1]},\boldsymbol{b}'^{[q+1]})\in\mathbb{R}^{m'_{q+1}\times m'_{\hat{q}}}\times\mathbb{R}^{m'_{q+1}}. \end{cases}$$

(ii) Layer linearization condition: For any $j\in[m_{\hat{q}}]$, there exists an affine subdomain (a_j,b_j) of σ associated with λ_j,μ_j such that the j-th component $(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}_{\boldsymbol{\theta}'}^{[q]}(\boldsymbol{x})+\boldsymbol{b}'^{[\hat{q}]})_j\in(a_j,b_j)$ for any $\boldsymbol{x}\in S_{\boldsymbol{x}}$.

(iii) Output preserving condition

$$\begin{cases} \boldsymbol{W}'^{[q+1]}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{W}'^{[\hat{q}]}=\boldsymbol{W}^{[q+1]},\\ \boldsymbol{W}'^{[q+1]}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{b}'^{[\hat{q}]}+\boldsymbol{W}'^{[q+1]}\boldsymbol{\mu}+\boldsymbol{b}'^{[q+1]}=\boldsymbol{b}^{[q+1]}, \end{cases}$$

where $\boldsymbol{\lambda}=[\lambda_1,\lambda_2,\cdots,\lambda_{m_{\hat{q}}}]^\top\in\mathbb{R}^{m'_{\hat{q}}}$, $\boldsymbol{\mu}=[\mu_1,\mu_2,\cdots,\mu_{m_{\hat{q}}}]^\top\in\mathbb{R}^{m'_{\hat{q}}}$, and $\mathrm{diag}(\boldsymbol{\lambda})$ *denotes the diagonal matrix formed by vector $\boldsymbol{\lambda}$.*

A.1 Existence of one-layer lifting

Lemma A.1 (Existence of One-Layer Lifting)**.** *Given data S, an* NN$(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* NN$'(\{m'_l\}, l\in\{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\})$*, the one-layer lifting $\mathcal{T}_S$ exists, i.e. $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$ is not empty for any parameter $\boldsymbol{\theta}_{\text{shal}}$ of* NN*.*

Proof. We prove this lemma by construction. From the definition of one-layer deeper, we know that

$$m'_1=m_1,\cdots,m'_q=m_q,m'_{\hat{q}}\geq\min\{m_q,m_{q+1}\},m'_{q+1}=m_{q+1},\cdots,m'_L=m_L.$$

Without loss of generality, we assume the width of the inserted layer $m'_{\hat{q}}$ is equal to $\min\{m_q,m_{q+1}\}$. Our construction can be easily extended to the case with a wider inserted layer by adding zero-neurons, i.e. neurons whose input and output weights are all zero.

For any parameter $\boldsymbol{\theta}_{\text{shal}}=(\boldsymbol{W}^{[1]},\boldsymbol{b}^{[1]},\cdots,\boldsymbol{W}^{[L]},\boldsymbol{b}^{[L]})$ of the NN, we construct a $\boldsymbol{\theta}'_{\text{deep}}$ in $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$ as follows. Since the activation function σ has a non-constant linear segment, there exists an affine subdomain (a,b) associated with $\lambda,\mu\in\mathbb{R}$ $(\lambda\neq 0)$ such that $\sigma(x)=\lambda x+\mu$ for $x\in(a,b)$. Let $[x_{\text{low}},x_{\text{up}}]\subseteq(a,b)$ $(x_{\text{low}}\neq x_{\text{up}})$ be a closed interval of the affine subdomain and $\boldsymbol{\lambda}=\lambda\mathbf{1}\in\mathbb{R}^{m_{\hat{q}}},\boldsymbol{\mu}=\mu\mathbf{1}\in\mathbb{R}^{m_{\hat{q}}}$, where $\mathbf{1}\in\mathbb{R}^{m_{\hat{q}}}$ is the all-ones vector. Now we discuss in two cases:

(1) $\min\{m_q,m_{q+1}\}=m_{q+1}$.

Denote training data by $S=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^{n}$ and

$$\begin{aligned}
&\tilde{\boldsymbol{f}}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}=\boldsymbol{W}^{[q+1]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}+\boldsymbol{b}^{[q+1]}\in\mathbb{R}^{m_{q+1}},\\
&x_{\min}=\min_{i\in[n],j\in[m_{q+1}]}\left\{\left(\tilde{\boldsymbol{f}}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i)\right)_j\right\},\\
&x_{\max}=\max_{i\in[n],j\in[m_{q+1}]}\left\{\left(\tilde{\boldsymbol{f}}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i)\right)_j\right\},
\end{aligned}$$

where $(\tilde{\boldsymbol{f}}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i))_j$ is the j-th component of $\tilde{\boldsymbol{f}}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i)$.

Now we transform the input range $[x_{\min},x_{\max}]$ into the affine subdomain $[a,b]$ of the activation function σ through an affine transformation. To this end, we further discuss in two cases:

Case 1: $x_{\min}\neq x_{\max}$.

Let

$$\begin{aligned}
&\xi=\frac{x_{\text{up}}-x_{\text{low}}}{x_{\max}-x_{\min}}\in\mathbb{R},\quad \boldsymbol{W}'^{[\hat{q}]}=\xi\boldsymbol{W}^{[q+1]}\in\mathbb{R}^{m_{\hat{q}}\times m_q},\\
&\boldsymbol{b}'^{[\hat{q}]}=\xi\boldsymbol{b}^{[q+1]}+(x_{\text{low}}-\xi x_{\min})\mathbf{1}\in\mathbb{R}^{m_{\hat{q}}},
\end{aligned}$$

where $\mathbf{1}$ is the all-ones vector. And then we let $\boldsymbol{W}'^{[q+1]} = \boldsymbol{I}_d/\lambda\xi \in \mathbb{R}^{m_{q+1}\times m_{\hat{q}}}$, where $\boldsymbol{I}_d$ is the identity matrix, and

$$\boldsymbol{b}'^{[q+1]} = \boldsymbol{b}^{[q+1]} - \boldsymbol{W}'^{[q+1]}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{b}'^{[\hat{q}]} - \boldsymbol{W}'^{[q+1]}\boldsymbol{\mu} \in \mathbb{R}^{m_{q+1}}.$$

Finally, we let

$$\boldsymbol{\theta}'_{\text{deep}} = \left(\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[q]}, \boldsymbol{b}^{[q]}, \boldsymbol{W}'^{[\hat{q}]}, \boldsymbol{b}'^{[\hat{q}]}, \boldsymbol{W}'^{[q+1]}\boldsymbol{b}'^{[q+1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]}\right).$$

Now we verify that $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$, i.e. $\boldsymbol{\theta}'_{\text{deep}}$ satisfies the three conditions of one-layer lifting. Firstly, by the construction of $\boldsymbol{\theta}'_{\text{deep}}$, the local-in-layer condition is satisfied automatically.

Next, for any $j \in [m_{\hat{q}}]$, there exists an affine subdomain (a,b) associated with λ, μ such that the j-th component

$$\begin{aligned}
&\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right)_j \\
=& \left(\xi\boldsymbol{W}^{[q+1]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \xi\boldsymbol{b}^{[q+1]} + (x_{\text{low}} - \xi x_{\min})\mathbf{1}\right)_j \\
=& \left(\xi\tilde{\boldsymbol{f}}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}} + (x_{\text{low}} - \xi x_{\min})\mathbf{1}\right)_j \in (a,b)
\end{aligned}$$

for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$. Thus, the layer linearization condition holds.

Finally, by direct calculation, the output preserving condition holds

$$\begin{cases}
\boldsymbol{W}'^{[q+1]}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{W}'^{[\hat{q}]} = \boldsymbol{W}^{[q+1]}, \\
\boldsymbol{W}'^{[q+1]}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{b}'^{[\hat{q}]} + \boldsymbol{W}'^{[q+1]}\boldsymbol{\mu} + \boldsymbol{b}'^{[q+1]} = \boldsymbol{b}^{[q+1]}.
\end{cases}$$

Collecting the above results, we prove that $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$, i.e. $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$ is not empty.

Case 2: $x_{\min} = x_{\max}$.

The layer linearization condition can be easily satisfied because the inputs to each neuron remain a constant. Therefore, by setting $\xi \neq 0$ to any nonzero constant, the above construction works for this case, i.e. the constructed $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$.

(2) $\min\{m_q, m_{q+1}\} = m_q$.

Denote training data by $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ and

$$\begin{aligned}
x_{\min} &= \min_{i\in[n], j\in[m_q]} \left\{\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i)\right)_j\right\}, \\
x_{\max} &= \max_{i\in[n], j\in[m_q]} \left\{\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i)\right)_j\right\},
\end{aligned}$$

where $(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i))_j$ is the j-th component of $\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}_i)$.

Also, we can transform the input range $[x_{\min}, x_{\max}]$ into the affine subdomain $[a,b]$ of the activation function σ through an affine transformation. To this end, we also further discuss in two cases:

Case 1: $x_{\min} \neq x_{\max}$.

Let

$$\xi = \frac{x_{\text{up}} - x_{\text{low}}}{x_{\max} - x_{\min}} \in \mathbb{R}, \quad \boldsymbol{W}'^{[\hat{q}]} = \xi \boldsymbol{I}_d \in \mathbb{R}^{m_{\hat{q}} \times m_q},$$
$$\boldsymbol{b}'^{[\hat{q}]} = (x_{\text{low}} - \xi x_{\min})\mathbf{1} \in \mathbb{R}^{m_{\hat{q}}}.$$

And then we let

$$\boldsymbol{W}'^{[q+1]} = \frac{1}{\lambda \xi} \boldsymbol{W}^{[q+1]} \in \mathbb{R}^{m_{q+1} \times m_{\hat{q}}},$$
$$\boldsymbol{b}'^{[q+1]} = \boldsymbol{b}^{[q+1]} - \boldsymbol{W}'^{[q+1]} \operatorname{diag}(\boldsymbol{\lambda}) \boldsymbol{b}'^{[\hat{q}]} - \boldsymbol{W}'^{[q+1]} \boldsymbol{\mu} \in \mathbb{R}^{m_{q+1}}.$$

Finally, we set

$$\boldsymbol{\theta}'_{\text{deep}} = \left(\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[q]}, \boldsymbol{b}^{[q]}, \boldsymbol{W}'^{[\hat{q}]}, \boldsymbol{b}'^{[\hat{q}]}, \boldsymbol{W}'^{[q+1]}, \boldsymbol{b}'^{[q+1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]}\right).$$

We can verify that $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$ similar to the previous Case 1.

Case 2: $x_{\min} = x_{\max}$.

By setting $\xi \neq 0$ to any nonzero constant, the above construction also works for this case, i.e. the constructed $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$.

Therefore, $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$ is non-empty for any $\boldsymbol{\theta}_{\text{shal}}$, i.e. one-layer lifting exists. □

A.2 Output preserving and criticality preserving

Now we prove the following lemma, of which Proposition A.1 is a direct consequence.

Lemma A.2 (Computation of Feature Vectors, Feature Gradients and Error Vectors)**.** *Given data S, consider an* $\text{NN}(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* $\text{NN}'(\{m'_l\}, l \in \{0,1,2,\ldots, q,\hat{q},q+1,\ldots,L\})$. *Let $\mathcal{T}_S$ denote the one-layer lifting and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN. *Then, for any lifted point $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$, the following conditions hold: There exist $\boldsymbol{\lambda}, \boldsymbol{\mu} \in \mathbb{R}^{m'_{\hat{q}}}$ such that for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$,*

(i) feature vectors in $\boldsymbol{F}_{\boldsymbol{\theta}'_{\text{deep}}}$

$$\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}), \quad l \in [L],$$
$$\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \operatorname{diag}(\boldsymbol{\lambda}) \left(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]} \right) + \boldsymbol{\mu},$$

(ii) feature gradients in $\boldsymbol{G}_{\boldsymbol{\theta}'_{\text{deep}}}$

$$\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{g}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}), \quad l \in [L],$$
$$\boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{\lambda},$$

(iii) error vectors in $\boldsymbol{Z}_{\boldsymbol{\theta}'_{\text{deep}}}$

$$\begin{aligned}
&\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{z}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}), \quad l \in [q-1] \cup [q+1:L],\\
&\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \left(\boldsymbol{W}'^{[q+1]}\right)^{\top}\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right),\\
&\boldsymbol{z}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \left(\boldsymbol{W}'^{[\hat{q}]}\right)^{\top}\left(\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{\lambda}\right).
\end{aligned}$$

Proof. (i) By the construction of $\boldsymbol{\theta}'_{\text{deep}}$, it is clear that $\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})$ for any $l \in [q]$. And by the definition of one-layer lifting, layer linearization condition is satisfied, i.e. for any $j \in [m_{\hat{q}}]$, there exists an affine subdomain (a_j, b_j) associated with λ_j, μ_j such that the j-th component $(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]})_j \in (a_j, b_j)$ for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$. Therefore, there exist $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_{m_{\hat{q}}}]^{\top} \in \mathbb{R}^{m'_{\hat{q}}}, \boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_{m_{\hat{q}}}]^{\top} \in \mathbb{R}^{m'_{\hat{q}}}$ such that for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$,

$$\begin{aligned}
\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right)\\
&= \boldsymbol{\lambda} \circ \left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right) + \boldsymbol{\mu}\\
&= \boldsymbol{\lambda} \circ \left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right) + \boldsymbol{\mu}\\
&= \text{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right) + \boldsymbol{\mu}.
\end{aligned}$$

By the forward propagation process and the output preserving condition of one-layer lifting, we have

$$\begin{aligned}
\boldsymbol{f}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{W}'^{[q+1]}\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[q+1]}\right)\\
&= \sigma\left(\boldsymbol{W}'^{[q+1]}\text{diag}(\boldsymbol{\lambda})\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{W}'^{[q+1]}\text{diag}(\boldsymbol{\lambda})\boldsymbol{b}'^{[\hat{q}]} + \boldsymbol{W}'^{[q+1]}\boldsymbol{\mu} + \boldsymbol{b}'^{[q+1]}\right)\\
&= \sigma\left(\boldsymbol{W}^{[q+1]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \boldsymbol{b}^{[q+1]}\right) = \boldsymbol{f}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}).
\end{aligned}$$

And by recursion, we have $\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \boldsymbol{f}^{[l]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})$ for $l \in [q+1:L]$.

(ii) By the continuity of the feature function, we know that for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$, there exists at least a neighborhood of $\boldsymbol{x}$ such that the layer linearization condition holds. Thus, we have

$$\boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) = \sigma^{(1)}\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right) = \boldsymbol{\lambda}.$$

And the results for feature gradients $\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}), l \in [L]$ can be recursively calculated in a similar way.

(iii) By the backpropagation and the above facts in (i), we have

$$z^{[L]}_{\theta'_{\text{deep}}}(x)=\nabla\ell\Big(f^{[L]}_{\theta'_{\text{deep}}}(x),y\Big)=\nabla\ell\Big(f^{[L]}_{\theta_{\text{shal}}}(x),y\Big)=z^{[L]}_{\theta_{\text{shal}}}(x).$$

So for $l\in[q+1:L]$, it is clear that

$$\begin{aligned}
&z^{[l]}_{\theta'_{\text{deep}}}(x)=z^{[l]}_{\theta_{\text{shal}}}(x),\\
&z^{[\hat{q}]}_{\theta'_{\text{deep}}}(x)=\big(W'^{[q+1]}\big)^\top\Big(z^{[q+1]}_{\theta_{\text{shal}}}(x)\circ g^{[q+1]}_{\theta_{\text{shal}}}(x)\Big).
\end{aligned}$$

Use the result in (ii), for $l=q$,

$$z^{[q]}_{\theta'_{\text{deep}}}(x)=\big(W'^{[\hat{q}]}\big)^\top\Big(z^{[\hat{q}]}_{\theta'_{\text{deep}}}(x)\circ g^{[\hat{q}]}_{\theta_{\text{deep}}}(x)\Big)=\big(W'^{[\hat{q}]}\big)^\top\Big(z^{[\hat{q}]}_{\theta'_{\text{deep}}}(x)\circ\lambda\Big).$$

Since

$$f^{[l]}_{\theta'_{\text{deep}}}(x)=f^{[l]}_{\theta_{\text{shal}}}(x),\quad g^{[l]}_{\theta'_{\text{deep}}}(x)=g^{[l]}_{\theta_{\text{shal}}}(x),\quad l\in[L],$$

we have

$$z^{[l]}_{\theta'_{\text{deep}}}(x)=z^{[l]}_{\theta_{\text{shal}}}(x),\quad l\in[q-1].$$

The proof is complete. □

Proposition A.1 (Network Properties Preserving). *Given data S, consider an* NN$(\{m_l\}_{l=0}^L)$ *and its one-layer deeper counterpart,* NN$'(\{m'_l\},l\in\{0,1,2,\dots,q,\hat{q},q+1,\dots,L\})$. *Let $\mathcal{T}_S$ denote the one-layer lifting and θ_{shal} be any parameter of* NN. *Then, for any lifted point $\theta'_{\text{deep}}\in\mathcal{T}_S(\theta_{\text{shal}})$, the following conditions hold:*

(i) outputs are preserved $f_{\theta'_{\text{deep}}}(x)=f_{\theta_{\text{shal}}}(x)$ for $x\in S_x$,

(ii) empirical risk is preserved $R_S(\theta'_{\text{deep}})=R_S(\theta_{\text{shal}})$,

(iii) the network representations are preserved for all layers

$$\text{span}\left\{\left\{\left(f^{[\hat{q}]}_{\theta'_{\text{deep}}}(X)\right)_j\right\}_{j\in[m'_{\hat{q}}]}\cup\{\mathbf{1}\}\right\}=\text{span}\left\{\left\{\left(f^{[q]}_{\theta_{\text{shal}}}(X)\right)_j\right\}_{j\in[m_q]}\cup\{\mathbf{1}\}\right\},$$

and for the other index $l\in[L]$,

$$\text{span}\left\{\left\{\left(f^{[l]}_{\theta'_{\text{deep}}}(X)\right)_j\right\}_{j\in[m'_l]}\cup\{\mathbf{1}\}\right\}=\text{span}\left\{\left\{\left(f^{[l]}_{\theta_{\text{shal}}}(X)\right)_j\right\}_{j\in[m_l]}\cup\{\mathbf{1}\}\right\},$$

where

$$f^{[l]}_\theta(X)=\Big[f^{[l]}_\theta(x_1),f^{[l]}_\theta(x_2),\cdots,f^{[l]}_\theta(x_n)\Big]^\top\in\mathbb{R}^{n\times m'_{\hat{q}}}$$

and $\mathbf{1}\in\mathbb{R}^n$ is the all-ones vector.

Proof. The properties (i) and (ii) are direct consequences of Lemma A.2.

(iii) It is clear that for $l\in[L]$

$$\operatorname{span}\left\{\left\{\left(\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m'_{\hat{q}}]}\cup\{\mathbf{1}\}\right\}=\operatorname{span}\left\{\left\{\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m_q]}\cup\{\mathbf{1}\}\right\}.$$

Since for any $\boldsymbol{x}\in S_{\boldsymbol{x}}$,

$$\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})=\boldsymbol{\lambda}\circ\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})+\boldsymbol{b}'^{[\hat{q}]}\right)+\boldsymbol{\mu},$$

we have

$$\operatorname{span}\left\{\left\{\left(\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m'_{\hat{q}}]}\cup\{\mathbf{1}\}\right\}=\operatorname{span}\left\{\left\{\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{X})\right)_j\right\}_{j\in[m_q]}\cup\{\mathbf{1}\}\right\}.$$

Thus, we finish the proof. □

Proposition A.2 (Criticality Preserving)**.** *Given data S, consider an* $\mathrm{NN}(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper counterpart,* $\mathrm{NN}'\left(\{m'_l\},l\in\{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\}\right)$*. Let $\mathcal{T}_S$ denote the one-layer lifting and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. If $\boldsymbol{\theta}_{\text{shal}}$ of* NN *satisfies $\nabla_{\boldsymbol{\theta}}R_S(\boldsymbol{\theta}_{\text{shal}})=\mathbf{0}$, then $\nabla_{\boldsymbol{\theta}'}R_S(\boldsymbol{\theta}'_{\text{deep}})=\mathbf{0}$ for any lifted point $\boldsymbol{\theta}'_{\text{deep}}\in\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$.*

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:

$$\begin{aligned}
\nabla_{\boldsymbol{W}^{[l]}}R_S(\boldsymbol{\theta})&=\nabla_{\boldsymbol{W}^{[l]}}\mathbb{E}_S\ell\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}),\boldsymbol{y}\right)=\mathbb{E}_S\left(\left(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x})\circ\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[l-1]}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^{\top}\right),\\
\nabla_{\boldsymbol{b}^{[l]}}R_S(\boldsymbol{\theta})&=\nabla_{\boldsymbol{b}^{[l]}}\mathbb{E}_S\ell\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}),\boldsymbol{y}\right)=\mathbb{E}_S\left(\boldsymbol{z}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x})\circ\boldsymbol{g}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x})\right).
\end{aligned}$$

Then we have for $l\neq q,\hat{q},q+1$,

$$\begin{aligned}
\nabla_{\boldsymbol{W}'^{[l]}}R_S(\boldsymbol{\theta}'_{\text{deep}})&=\nabla_{\boldsymbol{W}^{[l]}}R_S(\boldsymbol{\theta}'_{\text{deep}})=\nabla_{\boldsymbol{W}^{[l]}}R_S(\boldsymbol{\theta}_{\text{shal}})=\mathbf{0},\\
\nabla_{\boldsymbol{b}'^{[l]}}R_S(\boldsymbol{\theta}'_{\text{deep}})&=\nabla_{\boldsymbol{b}^{[l]}}R_S(\boldsymbol{\theta}'_{\text{deep}})=\nabla_{\boldsymbol{b}^{[l]}}R_S(\boldsymbol{\theta}_{\text{shal}})=\mathbf{0}.
\end{aligned}$$

Also, for $l=q+1$,

$$\begin{aligned}
\nabla_{\boldsymbol{W}'^{[q+1]}}R_S\left(\boldsymbol{\theta}'_{\text{deep}}\right)&=\mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\circ\boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)^{\top}\right)\\
&=\mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\circ\boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left[\sigma\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})+\boldsymbol{b}'^{[\hat{q}]}\right)\right]^{\top}\right)\\
&=\mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\circ\boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left(\boldsymbol{\lambda}\circ\left(\boldsymbol{W}'^{[\hat{q}]}\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})+\boldsymbol{b}'^{[\hat{q}]}\right)+\boldsymbol{\mu}\right)^{\top}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left(\operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]}\right) + \boldsymbol{\mu}\right)^{\top}\right) \\
&= \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)^{\top}\left(\operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[\hat{q}]}\right)^{\top}\right)\right) \\
&\quad + \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left(\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{b}'^{[\hat{q}]} + \boldsymbol{\mu}\right)^{\top}\right) = \mathbf{0}, \\
\nabla_{\boldsymbol{b}'^{[q+1]}} R_S\left(\boldsymbol{\theta}'_{\text{deep}}\right) &= \mathbb{E}_S\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) = \mathbb{E}_S\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right) = \mathbf{0}.
\end{aligned}
$$

For $l = \hat{q}$,

$$
\begin{aligned}
\nabla_{\boldsymbol{W}'^{[\hat{q}]}} R_S\left(\boldsymbol{\theta}'_{\text{deep}}\right) &= \mathbb{E}_S\left(\left(\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \mathbb{E}_S\left(\left(\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \mathbb{E}_S\left(\left(\operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[q+1]}\right)^{\top}\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\right)\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[q+1]}\right)^{\top}\mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)^{\top}\right) = \mathbf{0}, \\
\nabla_{\boldsymbol{b}'^{[\hat{q}]}} R_S\left(\boldsymbol{\theta}'_{\text{deep}}\right) &= \mathbb{E}_S\left(\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \\
&= \operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[q+1]}\right)^{\top}\mathbb{E}_S\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \\
&= \operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[q+1]}\right)^{\top}\mathbb{E}_S\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right) = \mathbf{0}.
\end{aligned}
$$

For $l = q$,

$$
\begin{aligned}
\nabla_{\boldsymbol{W}'^{[q]}} R_S\left(\boldsymbol{\theta}'_{\text{deep}}\right) &= \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q-1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \mathbb{E}_S\left(\left(\left(\boldsymbol{W}'^{[\hat{q}]}\right)^{\top}\left(\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q-1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \operatorname{diag}\left(\boldsymbol{\lambda}\right)\left(\boldsymbol{W}'^{[\hat{q}]}\right)^{\top}\left(\boldsymbol{W}'^{[q+1]}\right)^{\top} \\
&\quad \times \mathbb{E}_S\left(\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q-1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \left(\boldsymbol{W}^{[q+1]}\right)^{\top}\mathbb{E}_S\left(\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q-1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \left(\boldsymbol{W}^{[q+1]}\right)^{\top}\mathbb{E}_S\left(\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q-1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)^{\top}\right) \\
&= \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)\left(\boldsymbol{f}^{[q-1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right)^{\top}\right) = \mathbf{0},
\end{aligned}
$$

$$\begin{aligned}
\nabla_{\boldsymbol{b}'^{[q]}} R_S(\boldsymbol{\theta}'_{\text{deep}}) &= \mathbb{E}_S\left(\boldsymbol{z}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \\
&= \mathbb{E}_S\left((\boldsymbol{W}'^{[\hat{q}]})^\top \left(\boldsymbol{z}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[\hat{q}]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \\
&= \operatorname{diag}(\boldsymbol{\lambda})\left(\boldsymbol{W}'^{[\hat{q}]}\right)^\top \left(\boldsymbol{W}'^{[q+1]}\right)^\top \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \\
&= \left(\boldsymbol{W}^{[q+1]}\right)^\top \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}'_{\text{deep}}}(\boldsymbol{x})\right) \\
&= \left(\boldsymbol{W}^{[q+1]}\right)^\top \mathbb{E}_S\left(\left(\boldsymbol{z}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q+1]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right) \\
&= \mathbb{E}_S\left(\boldsymbol{z}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x}) \circ \boldsymbol{g}^{[q]}_{\boldsymbol{\theta}_{\text{shal}}}(\boldsymbol{x})\right) = \boldsymbol{0}.
\end{aligned}$$

Collecting all the above relations, we obtain that $\nabla_{\boldsymbol{\theta}'} R_S(\boldsymbol{\theta}'_{\text{deep}}) = \boldsymbol{0}$. □

A.3 Embedding principle in depth

Theorem A.1 (Embedding Principle in Depth)**.** *Given data S and an* $\mathrm{NN}'(\{m'_l\}_{l=0}^{L'})$*, for any parameter $\boldsymbol{\theta}_{\mathrm{c}}$ of any shallower* $\mathrm{NN}(\{m_l\}_{l=0}^{L})$ *satisfying $\nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}_{\mathrm{c}}) = \boldsymbol{0}$, there exists parameter $\boldsymbol{\theta}'_{\mathrm{c}}$ in the loss landscape of* $\mathrm{NN}'(\{m'_l\}_{l=0}^{L'})$ *satisfying the following conditions:*

(i) $f_{\boldsymbol{\theta}'_{\mathrm{c}}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}_{\mathrm{c}}}(\boldsymbol{x})$ *for* $\boldsymbol{x} \in S_{\boldsymbol{x}}$,

(ii) $\nabla_{\boldsymbol{\theta}'} R_S(\boldsymbol{\theta}'_{\mathrm{c}}) = \boldsymbol{0}$.

Proof. We prove this theorem by construction using the critical liftings. Let $J = L - L'$. The J-layer lifting $\mathcal{T}_S$ is the J-step composition of one-layer liftings, say $\mathcal{T}_S = \mathcal{T}_S^J \cdots \mathcal{T}_S^2 \mathcal{T}_S^1$. From Lemma A.1, we know one-layer lifting always exists, which leads to the existence of J-layer lifting $\mathcal{T}_S$, i.e. $\mathcal{T}_S(\boldsymbol{\theta}_c) \neq \emptyset$ for any $\boldsymbol{\theta}_c$. Now we prove by induction that J-layer lifting $\mathcal{T}_S$ satisfies the properties of output preserving and criticality preserving.

For $J = 1$, Propositions A.1 and A.2 show that the one-layer lifting satisfies the properties of output preserving and criticality preserving.

Assume that the $(J-1)$-layer lifting satisfies the properties of output preserving and criticality preserving, we want to show that so does the J-layer lifting.

From the induction hypothesis, we only need to show that if given two critical liftings $\mathcal{T}_S^1$ and $\mathcal{T}_S^2$, then $\mathcal{T}_S^2 \mathcal{T}_S^1$ also satisfies the properties of output preserving and criticality preserving.

(i) $\mathcal{T}_S^2 \mathcal{T}_S^1$ satisfies the property of output preserving:

Since $\mathcal{T}_S^1$ satisfies the property of output preserving, then for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$ and $\boldsymbol{\theta}' \in \mathcal{T}_S^1(\boldsymbol{\theta})$, $f_{\boldsymbol{\theta}'}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$. Similarly, for $\mathcal{T}_S^2$, we have for any $\boldsymbol{\theta}'' \in \mathcal{T}_S^2 \mathcal{T}_S^1(\boldsymbol{\theta})$, $f_{\boldsymbol{\theta}''}(\boldsymbol{x}) = f_{\boldsymbol{\theta}'}(\boldsymbol{x})$, hence $f_{\boldsymbol{\theta}''}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$.

(ii) $\mathcal{T}_2\mathcal{T}_1$ satisfies the property of criticality preserving:

Since $\boldsymbol{\theta}$ is a critical point of $R_S(\boldsymbol{\theta})$, for any $\boldsymbol{\theta}' \in \mathcal{T}_S^1(\boldsymbol{\theta}), \boldsymbol{\theta}'$ is also a critical point of $R_S(\boldsymbol{\theta})$. Similarly, for any $\boldsymbol{\theta}'' \in \mathcal{T}_S^2(\boldsymbol{\theta}'), \boldsymbol{\theta}''$ is also a critical point of $R_S(\boldsymbol{\theta})$, hence for any $\boldsymbol{\theta}'' \in \mathcal{T}_S^2\mathcal{T}_S^1(\boldsymbol{\theta}), \boldsymbol{\theta}''$ is also a critical point of $R_S(\boldsymbol{\theta})$.

Therefore, for any $\boldsymbol{\theta}_c' \in \mathcal{T}_S(\boldsymbol{\theta}_c)$, conditions (i) and (ii) are satisfied. □

A.4 Data dependency

Proposition A.3 (Data Dependency of Lifting)**.** *Given data S, S', consider an* $\mathrm{NN}(\{m_l\}_{l=0}^L)$ *and its deeper counterpart,* $\mathrm{NN}'(\{m_l'\}_{l=0}^{L'})$*. Let $\mathcal{T}_S$ and $\mathcal{T}_{S'}$ denote the respective critical liftings and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. If data $S' \subseteq S$, then $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}}) \subseteq \mathcal{T}_{S'}(\boldsymbol{\theta}_{\text{shal}})$.*

Proof. We first assume $\mathcal{T}_S$ is a one-layer lifting. If $S' \subseteq S$, i.e. dataset S' is a subset of dataset S, then we have $S_{\boldsymbol{x}}' \subseteq S_{\boldsymbol{x}}$. For any $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$, local-in-layer condition is satisfied regardless of input data. Regarding the data-dependent layer linearization condition, we have that for any $j \in [m_{\hat{q}}]$, there exists an affine subdomain (a_j, b_j) associated with λ_j, μ_j such that the j-th component $(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]})_j \in (a_j, b_j)$ for any $\boldsymbol{x} \in S_{\boldsymbol{x}}$, where $\boldsymbol{W}'^{[\hat{q}]}$ and $\boldsymbol{b}'^{[\hat{q}]}$ are weight and bias of $\boldsymbol{\theta}'_{\text{deep}}$ at layer $\hat{q}$. Since $S_{\boldsymbol{x}}' \subseteq S_{\boldsymbol{x}}$, for any $\boldsymbol{x} \in S_{\boldsymbol{x}}'$, naturally, $(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{deep}}}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]})_j \in (a_j, b_j)$, i.e. $\boldsymbol{\theta}'_{\text{deep}}$ satisfies layer linearization condition for S' with the same $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as for S. Therefore, $\boldsymbol{\theta}'_{\text{deep}}$ also satisfies the output preserving condition for S'. Then, we have $\boldsymbol{\theta}_{\text{deep}} \in \mathcal{T}_{S'}(\boldsymbol{\theta}_{\text{shal}})$, which leads to $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}}) \subseteq \mathcal{T}_{S'}(\boldsymbol{\theta}_{\text{shal}})$.

If $\mathcal{T}_S$ is a composition of critical liftings that satisfy this corollary, say $\mathcal{T}_S = \mathcal{T}_S^2\mathcal{T}_S^1$. We have for any $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}})$, there exists $\boldsymbol{\theta}_{\text{mid}} \in \mathcal{T}_S^1(\boldsymbol{\theta}_{\text{shal}})$ such that $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S^2(\boldsymbol{\theta}_{\text{mid}})$. Then $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_{S'}^2(\boldsymbol{\theta}_{\text{mid}})$ and $\boldsymbol{\theta}_{\text{mid}} \in \mathcal{T}_{S'}^1(\boldsymbol{\theta}_{\text{shal}})$. Therefore, $\boldsymbol{\theta}_{\text{deep}} \in \mathcal{T}_{S'}^2\mathcal{T}_{S'}^1(\boldsymbol{\theta}_{\text{shal}}) = \mathcal{T}_{S'}(\boldsymbol{\theta}_{\text{shal}})$, which leads to $\mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}}) \subseteq \mathcal{T}_{S'}(\boldsymbol{\theta}_{\text{shal}})$. □

Proposition A.4. *Given data S and an* $\mathrm{NN}'(\{m_l'\}_{l=0}^{L'})$*, for any parameter $\boldsymbol{\theta}_{\text{c}}$ of any shallower* $\mathrm{NN}(\{m_l\}_{l=0}^{L})$*, there exists parameter $\boldsymbol{\theta}_{\text{c}}'$ in the loss landscape of* $\mathrm{NN}'(\{m_l'\}_{l=0}^{L'})$ *satisfying that: For any $\boldsymbol{x}_i \in S_{\boldsymbol{x}}$, there exists a neighbourhood $N(\boldsymbol{x}_i)$ of $\boldsymbol{x}_i$ such that $\boldsymbol{f}_{\boldsymbol{\theta}_{\text{c}}'}(\boldsymbol{x}) = \boldsymbol{f}_{\boldsymbol{\theta}_{\text{c}}}(\boldsymbol{x})$ for any $\boldsymbol{x} \in N(\boldsymbol{x}_i)$.*

Proof. We only prove this result for one-layer lifting and similar to Theorem A.1, the result of multi-layer lifting can be easily obtained by induction. Let $\mathcal{T}_S$ be any one-layer lifting and $\boldsymbol{\theta}_{\text{c}}' \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{c}})$. By the definition of one-layer lifting, layer linearization condition is satisfied, i.e. for any $j \in [m_{\hat{q}}]$, there exists an affine subdomain (a_j, b_j) associated with λ_j, μ_j such that the j-th component $(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{c}}'}(\boldsymbol{x}_i) + \boldsymbol{b}'^{[\hat{q}]})_j \in (a_j, b_j)$ for any $\boldsymbol{x}_i \in S_{\boldsymbol{x}}$. Let

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}^{[q]}_{\boldsymbol{\theta}_{\text{c}}'}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]},$$

and

$$\varepsilon = \min\{\boldsymbol{g}(\boldsymbol{x}_i)_j - a_j, b_i - \boldsymbol{g}(\boldsymbol{x}_i)_j\}.$$

By the continuity of the function $\boldsymbol{g}(\boldsymbol{x})$, there exists a δ neighborhood $N_\delta(\boldsymbol{x}_i)$ such that $|\boldsymbol{g}(\boldsymbol{x}_i)_j - \boldsymbol{g}(\boldsymbol{x})_j| < \varepsilon$ for any $\boldsymbol{x} \in N_\delta(\boldsymbol{x}_i)$, which implies that $\boldsymbol{g}(\boldsymbol{x})_j \in (a_j, b_j)$. Therefore, the layer linearization condition indeed holds not only for each training input but also at least a neighbourhood of each training input.

Similar to Lemma A.2, by recursive we can get the NN output function is actually preserved over a broader area of input space including at least a neighbourhood of each training input. Hence, if the training dataset is sufficiently large and representative, then our lifting operator effectively preserves the generalization performance. □

Proposition A.5 (Positive and Negative Index of Inertia Preserving)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^{L})$*, and its deeper counterpart* NN$'(\{m'_l\}_{l=0}^{L'})$*. Let $\mathcal{T}_S$ denote the corresponding critical lifting and $\boldsymbol{\theta}_{\text{shal}}$ be any parameter of* NN*. Then, for any critical embedding $\mathcal{E}: \mathbb{R}^M \to \mathbb{R}^{M'}$ resulting from $\mathcal{T}_S$, denote $\boldsymbol{\theta}'_{\text{deep}} := \mathcal{E}(\boldsymbol{\theta}_{\text{shal}})$, the number of positive and negative eigenvalues of Hessian matrix $\boldsymbol{H}_S(\boldsymbol{\theta}'_{\text{deep}})$ equals the counterparts of $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}})$.*

Proof. (i) On one hand, because $\mathcal{E}$ is a critical embedding, therefore, in the neighborhood of $\boldsymbol{\theta}_{\text{shal}}$, there exists a full rank matrix $\boldsymbol{A} \in \mathbb{R}^{M' \times M}, \boldsymbol{c} \in \mathbb{R}^{M'}$ such that $\mathcal{E}(\boldsymbol{\theta}_{\text{shal}}) = \boldsymbol{A}\boldsymbol{\theta}_{\text{shal}} + \boldsymbol{c}$. By the output preserving property of $\mathcal{E}$, we have

$$R_S(\boldsymbol{\theta}_{\text{shal}}) \equiv R_S(\boldsymbol{\theta}_{\text{deep}}) \equiv R_S\big(\mathcal{E}(\boldsymbol{\theta}_{\text{shal}})\big) \equiv R_S(\boldsymbol{A}\boldsymbol{\theta}_{\text{shal}} + \boldsymbol{c}).$$

Hence,

$$\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{\theta}_{\text{shal}}) \equiv \nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} R_S(\boldsymbol{A}\boldsymbol{\theta}_{\text{shal}} + \boldsymbol{c}).$$

Then

$$\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}}) \equiv \boldsymbol{A}^\top \boldsymbol{H}_S(\boldsymbol{A}\boldsymbol{\theta}_{\text{shal}} + \boldsymbol{c})\boldsymbol{A}.$$

Given any $\boldsymbol{\theta}_{\text{shal}}$, if $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}})$ has k_1 negative eigenvalues $\{\lambda_j^{\text{neg}}\}_{j=1}^{k_1}$ with associated orthonormal eigenvectors $\{\boldsymbol{e}_j^{\text{neg}}\}_{j=1}^{k_1}$, then $\{\boldsymbol{A}\boldsymbol{e}_j^{\text{neg}}\}_{j=1}^{k_1}$ satisfies, for any $\boldsymbol{e}_j^{\text{neg}}$,

$$\big(\boldsymbol{A}\boldsymbol{e}_j^{\text{neg}}\big)^\top \boldsymbol{H}_S(\boldsymbol{A}\boldsymbol{\theta}_{\text{shal}} + \boldsymbol{c})\boldsymbol{A}\boldsymbol{e}_j^{\text{neg}} = \big(\boldsymbol{e}_j^{\text{neg}}\big)^\top \boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}})\boldsymbol{e}_j^{\text{neg}} = \lambda_j^{\text{neg}} < 0.$$

By full rankness of $\boldsymbol{A}$, we have

$$\dim\left(\operatorname{span}\left(\left\{\boldsymbol{A}\boldsymbol{e}_j^{\text{neg}}\right\}_{j=1}^{k_1}\right)\right) = k_1.$$

Thus, $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{deep}}) = \boldsymbol{H}_S(\boldsymbol{A}\boldsymbol{\theta}_{\text{shal}} + \boldsymbol{c})$ has at least k_1 negative eigenvalues.

(ii) On the other hand, for any lifted point $\boldsymbol{\theta}_{\text{deep}}$, by the definition of critical lifting, there exists a neighborhood of $\boldsymbol{\theta}_{\text{deep}}$ such that the layer linearization condition holds. Therefore, in the neighborhood of $\boldsymbol{\theta}_{\text{deep}}$, there exists a natural output-preserving projection operator $\mathcal{P}$ and a full rank matrix $\boldsymbol{B} \in \mathbb{R}^{M \times M'}, \boldsymbol{d} \in \mathbb{R}^M$ such that $\boldsymbol{\theta}_{\text{shal}} := \mathcal{P}(\boldsymbol{\theta}_{\text{deep}}) = \boldsymbol{B}\boldsymbol{\theta}_{\text{deep}} + \boldsymbol{d}$. By the output preserving property of $\mathcal{P}$, we have

$$R_S(\boldsymbol{\theta}_{\text{deep}}) \equiv R_S(\boldsymbol{\theta}_{\text{shal}}) \equiv R_S\big(\mathcal{P}(\boldsymbol{\theta}_{\text{deep}})\big) \equiv R_S(\boldsymbol{B}\boldsymbol{\theta}_{\text{deep}} + \boldsymbol{d}).$$

Hence,

$$\nabla_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}}R_S(\boldsymbol{\theta}_{\text{deep}}) \equiv \nabla_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}}R_S(\boldsymbol{B}\boldsymbol{\theta}_{\text{deep}}+\boldsymbol{d}).$$

Then

$$\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{deep}}) \equiv \boldsymbol{A}^\top \boldsymbol{H}_S(\boldsymbol{B}\boldsymbol{\theta}_{\text{deep}}+\boldsymbol{d})\boldsymbol{B}.$$

Given any $\boldsymbol{\theta}_{\text{deep}}$, if $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{deep}})$ has k_2 negative eigenvalues $\{\lambda_j^{\text{neg}}\}_{j=1}^{k_2}$ with associated orthonormal eigenvectors $\{\boldsymbol{e}_j^{\text{neg}}\}_{j=1}^{k_2}$, then $\{\boldsymbol{B}\boldsymbol{e}_j^{\text{neg}}\}_{j=1}^{k_2}$ satisfies, for any $\boldsymbol{e}_j^{\text{neg}}$,

$$\left(\boldsymbol{B}\boldsymbol{e}_j^{\text{neg}}\right)^\top \boldsymbol{H}_S(\boldsymbol{B}\boldsymbol{\theta}_{\text{deep}}+\boldsymbol{c})\boldsymbol{B}\boldsymbol{e}_j^{\text{neg}} = \left(\boldsymbol{e}_j^{\text{neg}}\right)^\top \boldsymbol{H}_S(\boldsymbol{\theta}_{\text{deep}})\boldsymbol{e}_j^{\text{neg}} = \lambda_j^{\text{neg}} < 0.$$

By full rankness of $\boldsymbol{B}$, we have

$$\dim\left(\text{span}\left(\left\{\boldsymbol{B}\boldsymbol{e}_j^{\text{neg}}\right\}_{j=1}^{k_2}\right)\right) = k_2.$$

Thus, $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}}) = \boldsymbol{H}_S(\boldsymbol{B}\boldsymbol{\theta}_{\text{deep}}+\boldsymbol{d})$ has at least k_2 negative eigenvalues.

Combining (i) and (ii), we have proved the number of negative eigenvalues of Hessian matrix $\boldsymbol{H}_S(\boldsymbol{\theta}'_{\text{deep}})$ equals the counterparts of $\boldsymbol{H}_S(\boldsymbol{\theta}_{\text{shal}})$. Similarly, we can prove this result for the number of positive eigenvalues. □

Corollary A.1 (Incremental Degeneracy of Critical Point Through Lifting)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^{L})$*, and its deeper counterpart* NN$'(\{m'_l\}_{l=0}^{L'})$*. Let $\mathcal{T}_S$ denote the corresponding critical lifting and $\boldsymbol{\theta}_{\text{shal}} \in \mathbb{R}^M$ be a critical point of* NN*. Then, any lifted point $\boldsymbol{\theta}'_{\text{deep}} \in \mathcal{T}_S(\boldsymbol{\theta}_{\text{shal}}) \subseteq \mathbb{R}^{M'}$ possesses $M'-M$ additional degrees of degeneracy in comparison to $\boldsymbol{\theta}_{\text{shal}}$.*

Proof. Given that deeper networks have more parameters, it follows from Proposition A.5 that the lifted critical point $\boldsymbol{\theta}_{\text{deep}}$ exhibits $M'-M$ extra degrees of degeneracy compared to $\boldsymbol{\theta}_{\text{shal}}$. □

A.5 One-layer residual lifting

We give the rigorous definition of one-layer residual lifting, which is very similar to one-layer lifting. The only difference is that there is one more item in the output preserving condition due to the skip connection (see Fig. 9 for illustration).

Definition A.2 (One-Layer Residual Lifting)**.** *Given data S, consider an* NN$(\{m_l\}_{l=0}^{L})$ *and its one-layer deeper residual counterpart,* NN$'(\{m'_l\}, l \in \{0,1,2,\ldots,q,\hat{q},q+1,\ldots,L\})$*, which has a skip connection at the $\hat{q}$-th layer. The one-layer residual lifting, denoted as $\mathcal{T}_S$, is a function that transforms any parameter $\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \cdots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]})$ of* NN *into a set $\mathcal{M}$ within the parameter space of* NN$'$*. Formally, $\mathcal{M}$ (where $\mathcal{M} := \mathcal{T}_S(\boldsymbol{\theta})$) represents a collection of all possible parameters $\boldsymbol{\theta}'$ of* NN$'$ *that satisfying the following three conditions:*

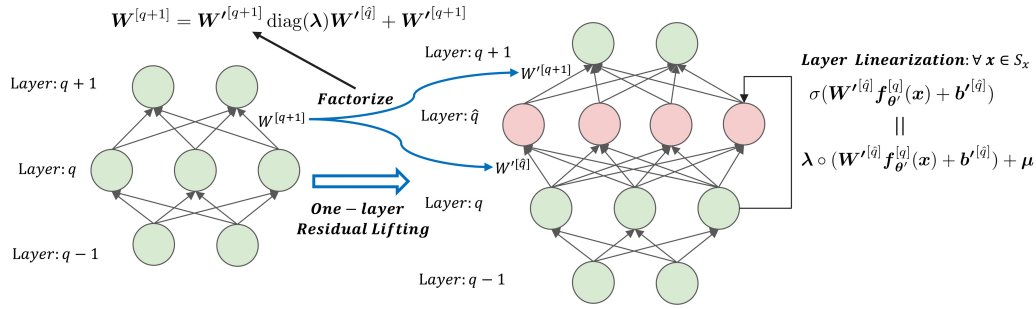


Figure 9: Illustration of one-layer residual lifting. The pink layer is inserted into the left network to get the right network. The input parameters $W'^{[\hat{q}]}$ and output parameters $W'^{[q+1]}$ of the inserted layer are obtained by factorizing the input parameters $W^{[q+1]}$ of $(q+1)$-th layer in the left network to satisfy layer linearization and output preserving conditions.

(i) Local-in-layer condition: Weights of each layer in NN' *are inherited from* NN *except for layer* $\hat{q}$ *and* $q+1$*, i.e.*

$$\begin{cases} \boldsymbol{\theta}'|_l = \boldsymbol{\theta}|_l \quad for \quad l \in [q] \cup [q+2:L], \\ \boldsymbol{\theta}'|_{\hat{q}} = (\boldsymbol{W}'^{[\hat{q}]}, \boldsymbol{b}'^{[\hat{q}]}) \in \mathbb{R}^{m'_{\hat{q}} \times m'_{q-1}} \times \mathbb{R}^{m'_{\hat{q}}}, \\ \boldsymbol{\theta}'|_{q+1} = (\boldsymbol{W}'^{[q+1]}, \boldsymbol{b}'^{[q+1]}) \in \mathbb{R}^{m'_{q+1} \times m'_{\hat{q}}} \times \mathbb{R}^{m'_{q+1}}. \end{cases}$$

(ii) Layer linearization condition: For any $j \in [m_{\hat{q}}]$*, there exists an affine subdomain* (a_j, b_j) *of* σ *associated with* λ_j, μ_j *such that the j-th component* $(\boldsymbol{W}'^{[\hat{q}]} \boldsymbol{f}_{\boldsymbol{\theta}'}^{[q]}(\boldsymbol{x}) + \boldsymbol{b}'^{[\hat{q}]})_j \in (a_j, b_j)$ *for any* $\boldsymbol{x} \in S_{\boldsymbol{x}}$*.*

(iii) Output preserving condition

$$\begin{cases} \boldsymbol{W}'^{[q+1]}\,\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{W}'^{[\hat{q}]} + \boldsymbol{W}'^{[q+1]} = \boldsymbol{W}^{[q+1]}, \\ \boldsymbol{W}'^{[q+1]}\,\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{b}'^{[\hat{q}]} + \boldsymbol{W}'^{[q+1]}\boldsymbol{\mu} + \boldsymbol{b}'^{[q+1]} = \boldsymbol{b}^{[q+1]}, \end{cases}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_{m_{\hat{q}}}]^\top \in \mathbb{R}^{m'_{\hat{q}}}, \boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_{m_{\hat{q}}}]^\top \in \mathbb{R}^{m'_{\hat{q}}}$*, and* $\mathrm{diag}(\boldsymbol{\lambda})$ *denotes the diagonal matrix formed by vector* $\boldsymbol{\lambda}$*.*

As with one-layer lifting, the properties of output preserving and criticality preserving as well as data-dependency are the same for one-layer residual lifting.

A.6 Centered kernel alignment

Consider $\boldsymbol{X} \in \mathbb{R}^{n \times m_1}$ and $\boldsymbol{Y} \in \mathbb{R}^{n \times m_2}$, which represent two layers each containing m_1 and m_2 neurons respectively, mapped to the identical set of n instances. The Gram matrices, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^\top$ and $\boldsymbol{L} = \boldsymbol{Y}\boldsymbol{Y}^\top$, are $n \times n$ in dimension and each of their elements signifies the similarity between two instances based on the representations in $\boldsymbol{X}$ or $\boldsymbol{Y}$.

The centering matrix is given by $\boldsymbol{H}=\boldsymbol{I}_n-\mathbf{1}\mathbf{1}^\top/n$. Consequently, the matrices $\boldsymbol{K}'=\boldsymbol{HKH}$ and $\boldsymbol{L}'=\boldsymbol{HLH}$ correspond to similarity matrices where column and row means have been subtracted.

The Hilbert-Schmidt independence criterion (HSIC) quantifies the similarity of these centered similarity matrices by converting them into vectors and computing the dot product between these vectors, $\mathrm{HSIC}_0(\boldsymbol{K},\boldsymbol{L})=\mathrm{vec}(\boldsymbol{K}')\cdot\mathrm{vec}(\boldsymbol{L}')/(n-1)^2$. HSIC remains invariant under orthogonal transformations of the representations and, consequently, under permutation of neurons, but lacks invariance to scaling of the original representations.

Centered kernel alignment (CKA) is used to further normalize HSIC to yield a similarity metric in the range of 0 to 1 that remains invariant to isotropic scaling

$$\mathrm{CKA}(\boldsymbol{K},\boldsymbol{L})=\frac{\mathrm{HSIC}_0(\boldsymbol{K},\boldsymbol{L})}{\sqrt{\mathrm{HSIC}_0(\boldsymbol{K},\boldsymbol{K})\mathrm{HSIC}_0(\boldsymbol{L},\boldsymbol{L})}}.$$

The research by [15] demonstrated that linear CKA is a reliable measure for identifying architecturally corresponding layers when measured between layers of architecturally identical networks that were trained from distinct random initializations. Indeed, linear CKA reflects the degree of linear correlation between representations across layers.

Proposition A.6 (CKA and Layer Linearization)**.** *Let $\boldsymbol{X}\in\mathbb{R}^{n\times m_1}$ and $\boldsymbol{Y}\in\mathbb{R}^{n\times m_2}$ contain representations of two layers, one with m_1 neurons and another m_2 neurons, to the same set of n examples. If the linear CKA between the two layers equals 1, then there exists $\boldsymbol{W}\in\mathbb{R}^{m_1\times m_2}$, $\boldsymbol{b}\in\mathbb{R}^{1\times m_2}$ such that*

$$\boldsymbol{Y}=\boldsymbol{XW}+\boldsymbol{b}.$$

Proof. Denote $\boldsymbol{X}'=\boldsymbol{HX},\boldsymbol{Y}'=\boldsymbol{HY}$, where $\boldsymbol{H}=\boldsymbol{I}_n-\mathbf{1}\mathbf{1}^\top/n$. Then $\boldsymbol{K}'=\boldsymbol{X}'\boldsymbol{X}'^\top,\boldsymbol{L}'=\boldsymbol{Y}'\boldsymbol{Y}'^\top$. Notice that

$$\left\langle\mathrm{vec}(\boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}}),\mathrm{vec}(\boldsymbol{Y}'\boldsymbol{Y}'^{\mathrm{T}})\right\rangle=\mathrm{tr}(\boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}}\boldsymbol{Y}'\boldsymbol{Y}'^{\mathrm{T}})=\mathrm{tr}(\boldsymbol{X}'^{\mathrm{T}}\boldsymbol{Y}'\boldsymbol{Y}'^{\mathrm{T}}\boldsymbol{X}')=\|\boldsymbol{Y}'^{\mathrm{T}}\boldsymbol{X}'\|_{\mathrm{F}}^2.$$

Linear CKA is equivalent to the cosine similarity

$$\begin{aligned}\mathrm{CKA}(\boldsymbol{XX}^T,\boldsymbol{YY}^T)&=\frac{\|\boldsymbol{Y}'^{\mathrm{T}}\boldsymbol{X}'\|_{\mathrm{F}}^2}{\|\boldsymbol{X}'^{\mathrm{T}}\boldsymbol{X}'\|_{\mathrm{F}}\|\boldsymbol{Y}'^{\mathrm{T}}\boldsymbol{Y}'\|_{\mathrm{F}}}\\&=\frac{\left\langle\mathrm{vec}(\boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}}),\mathrm{vec}(\boldsymbol{Y}'\boldsymbol{Y}'^{\mathrm{T}})\right\rangle}{\sqrt{\left\langle\mathrm{vec}(\boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}}),\mathrm{vec}(\boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}})\right\rangle}\sqrt{\left\langle\mathrm{vec}(\boldsymbol{Y}'\boldsymbol{Y}'^{\mathrm{T}}),\mathrm{vec}(\boldsymbol{Y}'\boldsymbol{Y}'^{\mathrm{T}})\right\rangle}}.\end{aligned}$$

If $\mathrm{CKA}(\boldsymbol{XX}^T,\boldsymbol{YY}^T)=1$, then there exists $\alpha\geq 0$ such that

$$\boldsymbol{Y}'\boldsymbol{Y}'^\top=\alpha\boldsymbol{X}'\boldsymbol{X}'^\top.$$

From this, we can conclude $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ share the same column space. Therefore, there exists a matrix $\boldsymbol{W}\in\mathbb{R}^{m_1\times m_2}$ such that $\boldsymbol{Y}'=\boldsymbol{X}'\boldsymbol{W}$. As $\boldsymbol{X}'=\boldsymbol{X}-\mathbf{1}\mathbf{1}^T\boldsymbol{X}/n$ and $\boldsymbol{Y}'=\boldsymbol{Y}-\mathbf{1}\mathbf{1}^T\boldsymbol{Y}/n$, we can write $\boldsymbol{Y}'=\boldsymbol{X}'\boldsymbol{W}$ as

$$\boldsymbol{Y}-\frac{1}{n}\mathbf{1}\mathbf{1}^T\boldsymbol{Y}=\boldsymbol{XW}-\frac{1}{n}\mathbf{1}\mathbf{1}^T\boldsymbol{XW}.$$

Denote

$$\boldsymbol{b}=\frac{1}{n}\mathbf{1}^{\top}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{W})\in\mathbb{R}^{1\times m_2},$$

then we have

$$\boldsymbol{Y}=\boldsymbol{X}\boldsymbol{W}+\boldsymbol{b},$$

which completes the proof. □

Appendix B. Supplementary experiments

In this section, we present the supplementary experiments mentioned in the main text.

Dependence of layer linearization on initialization scale and training data size. The layer linearization of a network is influenced by its initialization scale and the size of the training data. When initialized with a small enough scale, the network is likely to operate in the linear region during the early stages of training, leads to often encounter the lifted critical point. To investigate the impact of initialization scale on layer linearization, we train a 10-layer network with different initializations on the Fashion-MNIST dataset and measure the extent of linearization of each hidden layer post-training. As depicted in Fig. 12(a), with increasing initialization scale from left to right, the degree of nonlinearity within each hidden layer also rises. Additionally, as shown in Fig. 12(b), for a fixed initialization scale, the degree of nonlinearity across the network's hidden layers increases as the size of the training dataset expands from 500 to 5,000 to 50,000. This finding suggest that larger dataset adds to the difficulty of layer linearization and potentially facilitate a reduction in critical manifolds, thereby enhancing optimization. This is further corroborated by our experiment on simpler datasets, as depicted in Fig. 7 in the main text.

Appendix C. Details of experiments

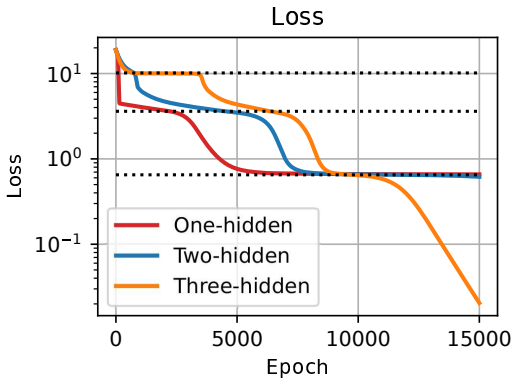
For the experiment of Iris dataset (Figs. 1(a) and 1(b)), we use ReLU as the activation function and the mean square error (MSE) as the loss function. We use full-batch gradient descent with learning rate 0.001 to train NNs for 100000 epochs. The width is 50 for each hidden layer. The initial distribution of all parameters follows a Gaussian distribution with a mean of 0 and a variance of 0.07. For the MNIST dataset shown in Figs. 1(c) and 1(d), we randomly select 500 images to constitute the training set, employing full batch gradient training, with MSE serving as the loss function. Remark that, the phenomenon in Fig. 1 are similar for different activation functions.

For the 1-D experiments in Figs. 3, 6, 7, and 11, we use tanh as the activation function and MSE as the loss function. We use full-batch gradient descent with learning rate 0.01 to train NNs with width 50 for each hidden layer. The initial distribution of all parameters follows a Gaussian distribution with a mean of 0 and a variance of 0.01.

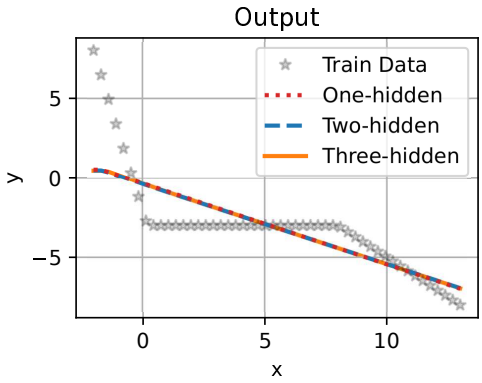
For the experiment of MNIST classification (Fig. 8), we use tanh as the activation function and the cross-entropy as the loss function. We use stochastic gradient descent with batch size 1000 and learning rate 0.001 to train NNs for 100 epochs. The width is 50 for each hidden layer. The initial distribution of all parameters follows a Gaussian distribution with a mean of 0 and a variance of 0.05.

For the experiment of Fashion-MNIST classification (Fig. 12), we use tanh as the activation function and the cross-entropy as the loss function. We use stochastic gradient descent with batch size 512 and learning rate 0.01 to train NNs for 50 epochs. The width is 100 for each hidden layer.

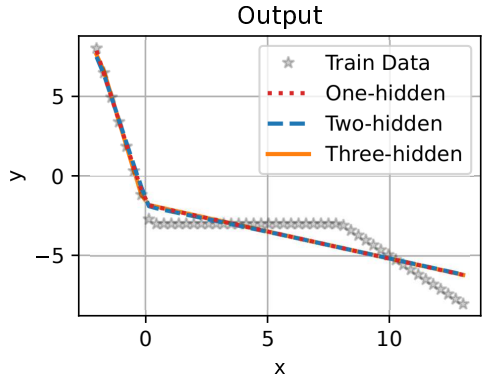
For the 1-D experiments in Fig. 10, we use ReLU as the activation function and MSE as the loss function. We use full-batch gradient descent with learning rate 0.001 to train NNs with width 50 for each hidden layer. The initial distribution of all parameters follows a Gaussian distribution with a mean of 0 and a variance of 0.005.



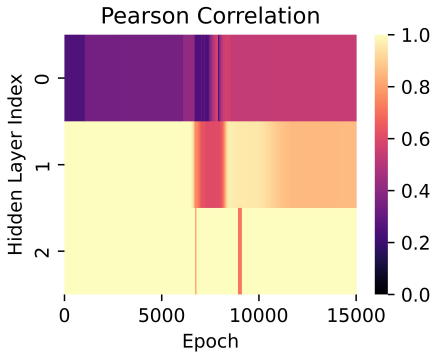
(a) Training loss



(b) Output corresponding to the second dotted line in (a)

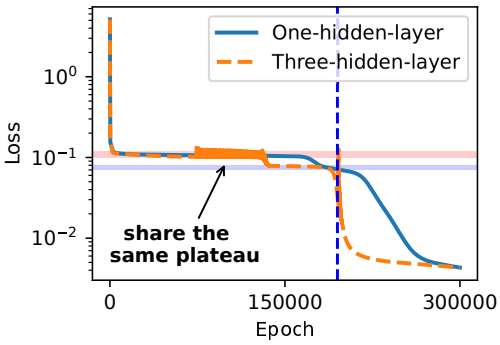


(c) Output corresponding to the third dotted line in (a)

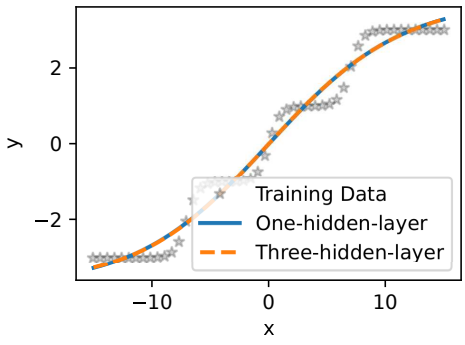


(d) Evolution of MPC

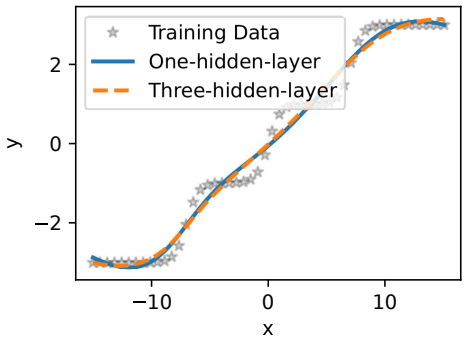
Figure 10: Deep ReLU neural networks encounter lifted critical points during training. (a) The training loss trajectory for ReLU NN of different depths with 50 neurons in each hidden layer on training data in (b). (b, c) The output functions of NNs with different depths at the same loss values indicated by (b) the second horizontal dotted line or (c) the third horizontal dotted line in (a). (d) The extent of layer linearization for all hidden layers during the training process of three-hidden-layer NN.



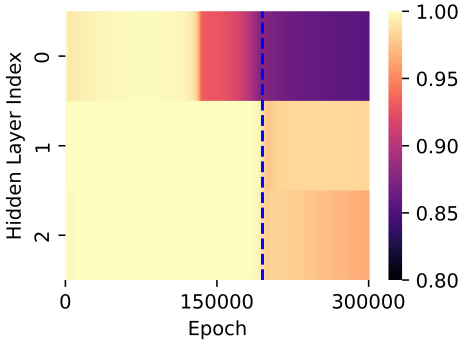
(a) Training loss



(b) Output corresponding to the red span in (a)

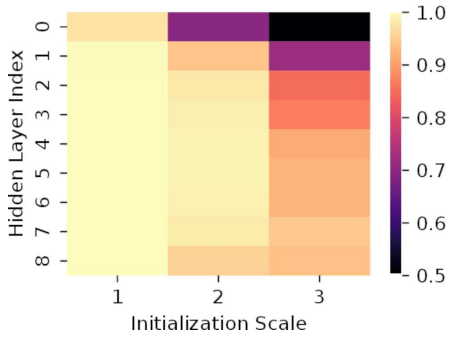


(c) Output corresponding to the blue span in (a)

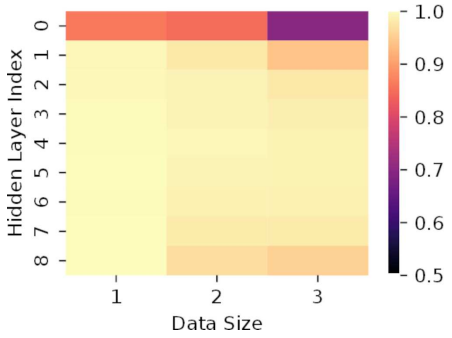


(d) Evolution of MPC

Figure 11: Deep residual-connected neural networks encounter lifted critical points during training. (a) The training loss for NNs of different depths with 50 neurons in each hidden layer for training data in (b). (b, c) The output functions of NNs with different depths at the same loss values indicated by (b) the first horizontal dotted line or (c) the second horizontal dotted line. (d) The extent of layer linearization for all hidden layers during the training process of three-hidden-layer residual NN.



(a) Initialization scale



(b) Training data size

Figure 12: Dependence of layer linearization on initialization scale and training data size. (a) The extent of layer linearization across all hidden layers corresponding to initialization scales drawn from a Gaussian distribution with a mean of 0 and standard deviations of 0.003, and 0.005, 0.02, arranged from left to right. (b) The extent of layer linearization across all hidden layers in relation to training data sizes of 500, 5,000, and 50,000, arranged from left to right.

For the experiments in Fig. 5, we employ a comprehensive methodology to ascertain the eigenvalues of the Hessian matrix at empirical critical points, comprising the following steps:

(1) Initially, we approximate the probable interval of critical points by observing regions where the loss diminishes very slowly. We then choose the point with the least parameter derivative (using the L_1 norm) as our empirical critical point. At this empirical critical point, the L_1 norm of the derivative of the loss function hovers around 10^{-4}, which is acceptably small.

(2) To accurately determine the eigenvalues of a Hessian matrix with a large condition number, we perform 100 random orthogonal similarity transformations on the matrix. We derive a more reliable set of eigenvalues by averaging the outcomes from these 100 trials.

(3) For a clearer distinction between significant and non-significant eigenvalues, we identify locations where there are evident gaps in eigenvalue magnitudes to differentiate between zero and non-zero eigenvalues.

(4) To further ensure that the empirical degeneracy is valid, we meticulously examined the state of each neuron and the effective rank of the Hessian matrix. For instance, in Fig. 5(a), for the ReLU network with a single hidden layer containing only two neurons, one of the neurons remains inactive throughout the training dataset. Consequently, the eigendirections associated with its parameters should be null. We observed that the effective rank of the Hessian matrix is 3, implying that there indeed are three non-zero eigenvalues. This aligns with the number of non-zero eigenvalues suggested by the evident gaps identified in step (3).

For activation functions with strong nonlinearity near zero (e.g. ReLU), we first removes those "zero-neurons" whose input and output weights are reasonably small to avoid their interference to the measure of layer linearization.

Remark that, although Figs. 1, 3 and 4 are case studies each based on a random trial, similar phenomenon can be easily observed as long as the initialization variance is properly small, i.e. far away from the linear/kernel/NTK regime.

Acknowledgments

This work is sponsored by the National Key R&D Program of China (Grant No. 2022YFA1008200) (T. Luo, Z. Xu and Y. Zhang), by the National Natural Science Foundation of China (Grant Nos. 92270001 (Z. Xu), 12101402 (Y. Zhang), 12371511 (Z. Xu), 12101401 (T. Luo)), by the Shanghai Municipal Science and Technology (Key Project No. 22JC1401500 (T. Luo)), by the Shanghai Municipal of Science and Technology (Grant No. 20JC1419500 (Y. Zhang)), by the Lingang Laboratory (Grant No. LG-QS-202202-08 (Y. Zhang)), by the Shanghai Municipal of Science and Technology (Major Project No.

2021SHZDZX0102), by the HPC of School of Mathematical Sciences and the Student Innovation Center, and the Siyuan-1 cluster supported by the Center for High Performance Computing at Shanghai Jiao Tong University, Key Laboratory of Marine Intelligent Equipment and System, Ministry of Education, P.R. China.

References

[1] S. Arora, N. Cohen, and E. Hazan, *On the optimization of deep networks: Implicit acceleration by overparameterization*, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 80:244–253, 2018.

[2] Z. Cai, J. Chen, M. Liu, and X. Liu, *Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs*, J. Comput. Phys., 420:109707, 2020.

[3] P. Cheridito, A. Jentzen, and F. Rossmannek, *Landscape analysis for shallow neural networks: Complete classification of critical points for affine target functions*, arXiv:2103.10922, 2021.

[4] R. Collobert and J. Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, in: Proceedings of the 25th International Conference on Machine learning, ACM, 160–167, 2008.

[5] S. Du and J. Lee, *On the power of over-parametrization in neural networks with quadratic activation*, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 80:1329–1338, 2018.

[6] W. E and Q. Wang, *Exponential convergence of the deep neural network approximationfor analytic functions*, Sci. China Math., 61:1733–1740, 2018.

[7] R. Eldan and O. Shamir, *The power of depth for feedforward neural networks*, in: 29th Annual Conference on Learning Theory, PMLR, Vol. 49, 907–940, 2016.

[8] K. Fukumizu and S.-i. Amari, *Local minima and plateaus in hierarchical structures of multilayer perceptrons*, Neural Netw., 13(3):317–327, 2000.

[9] K. Fukumizu, S. Yamaguchi, Y.-i. Mototake, and M. Tanaka, *Semi-flat minima and saddle points by embedding neural networks to overparameterization*, in: Advances in Neural Information Processing Systems, Vol. 32, 13868–13876, 2019.

[10] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci. USA, 115(34):8505–8510, 2018.

[11] H. He, G. Huang, and Y. Yuan, *Asymmetric valleys: Beyond sharp and flat local minima*, arXiv: 1902.00744, 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[13] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in: Proceedings of the 32nd International Conference on Machine Learning, PMLR, 37:448–456, 2015.

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

[15] S. R. Kudugunta, A. Bapna, I. Caswell, N. Arivazhagan, and O. Firat, *Investigating multilingual MNT representations at scale*, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 1565–1575, 2019.

[16] T. Nguyen, M. Raghu, and S. Kornblith, *Do wide and deep networks learn the same things? Uncovering how neural network representations vary with width and depth*, in: International Conference on Learning Representations, 2021.
[17] N. Rahaman, D. Arpit, A. Baratin, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of deep neural networks*, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 97:5301–5310, 2019.
[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, *XNOR-Net: Imagenet classification using binary convolutional neural networks*, in: Proceedings of the 14th European Conference on Computer Vision, Springer, 525–542, 2016.
[19] L. Sagun, L. Bottou, and Y. LeCun, *Eigenvalues of the Hessian in deep learning: Singularity and beyond*, arXiv:1611.07476, 2016.
[20] B. Simsek, F. Ged, A. Jacot, F. Spadaro, C. Hongler, W. Gerstner, and J. Brea, *Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances*, in: Proceedings of the 38th International Conference on Machine Learning, PMLR, 139:9722–9732, 2021.
[21] I. Skorokhodov and M. Burtsev, *Loss landscape sightseeing with multi-point optimization*, arXiv: 1910.03867, 2019.
[22] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, *Theoretical insights into the optimization landscape of over-parameterized shallow neural networks*, IEEE Trans. Inf. Theory, 65(2):742–769, 2019.
[23] M. Telgarsky, *Benefits of depth in neural networks*, in: 29th Annual Conference on Learning Theory, PMLR, 49:1517–1539, 2016.
[24] Z. Wang and Z. Zhang, *A mesh-free method for interface problems using the deep learning approach*, J. Comput. Phys., 400:108963, 2020.
[25] L. Wu, Z. Zhu, and W. E, *Towards understanding generalization of deep learning: Perspective of loss landscapes*, arXiv:1706.10239, 2017.
[26] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, *Frequency principle: Fourier analysis sheds light on deep neural networks*, Commun. Comput. Phys., 28(5):1746–1767, 2020.
[27] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, *Training behavior of deep neural network in frequency domain*, in: International Conference on Neural Information Processing, 264–274, 2019.
[28] Z.-Q. J. Xu and H. Zhou, *Deep frequency principle towards understanding why deeper learning is faster*, in: Proceedings of the AAAI Conference on Artificial Intelligence, 35(12):10541–10550, 2021.
[29] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, *Understanding deep learning requires rethinking generalization*, in: 5th International Conference on Learning Representations, ICLR 2017, 2017.
[30] Y. Zhang, Y. Li, Z. Zhang, T. Luo, and Z. J. Xu, *Embedding principle: A hierarchical structure of loss landscape of deep neural networks*, J. Mach. Learn., 1:60–113, 2022.
[31] Y. Zhang, T. Luo, Z. Ma, and Z.-Q. J. Xu, *A linear frequency principle model to understand the absence of overfitting in neural networks*, Chinese Phys. Lett., 38:038701, 2021.
[32] Y. Zhang, Z. Zhang, T. Luo, and Z. J. Xu, *Embedding principle of loss landscape of deep neural networks*, in: Advances in Neural Information Processing Systems, 34:14848–14859, 2021.