Rank Diminishing in Deep Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

The rank of neural networks measures information flowing across layers. It is an instance of a key structural condition that applies across broad domains of machine learning. In particular, the assumption of low-rank feature representations leads to algorithmic developments in many architectures. For neural networks, however, the intrinsic mechanism that yields low-rank structures remains vague and unclear. To fill this gap, we perform a rigorous study on the behavior of network rank, focusing particularly on the notion of rank deficiency. We theoretically establish a universal monotonic decreasing property of network rank from the basic rules of differential and algebraic composition, and uncover rank deficiency of network blocks and deep function coupling. By virtue of our numerical tools, we provide the first empirical analysis of the per-layer behavior of network rank in practical settings, *i.e.*, ResNets, deep MLPs, and Transformers on ImageNet. These empirical results are in direct accord with our theory. Furthermore, we reveal a novel phenomenon of independence deficit caused by the rank deficiency of deep networks, where classification confidence of a given category can be linearly decided by the confidence of a handful of other categories. The theoretical results of this work, together with the empirical findings, may advance understanding of the inherent principles of deep neural networks.

1 Introduction

In mathematics, the rank of a smooth function measures the volume of independent information captured by the function [23]. Deep neural networks are highly smooth functions, thus the rank of a network has long been an essential concept in machine learning that underlies many tasks such as information compression [53, 63, 38, 61, 54], network pruning [33, 62, 6, 27, 10], data mining [7, 26, 11, 64, 19, 30], computer vision [66, 65, 32, 28, 30, 67], and natural language processing [9, 29, 8, 12]. Numerous methods are either designed to utilize the mathematical property of network ranks, or are derived from an assumption that low-rank structures are to be preferred.

Yet a rigorous investigation to the behavior of rank of general networks, combining both theoretical and empirical arguments, is still absent in current research, weakening our confidence in the being able to predict performance. To the best of our knowledge, there are only a few previous works discussing the rank behavior of specific network architectures, like attention blocks [15] and BatchNorms [13, 5] in pure MLP structures. The empirical validation of those methods are also limited to shallow networks, specific architectures, or merely the final layers of deep networks, leaving the global behavior of general deep neural networks mysterious due to prohibitive space-time complexity for measuring them. Rigorous work on network rank that combines both strong theoretical and empirical evidence would have significant implications.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we make several contributions towards this challenging goal. We find that the two essential ingredients of deep learning, the chain rules of differential operators and matrix multiplications, are enough to establish a universal principle—that network rank decreases monotonically with the depth of networks. Two factors further enhance the speed of decreasing: a) the explicit rank deficiency of many frequently used network modules, and b) an intrinsic potential of spectrum centralization enforced by the nature of coupling of massive composite functions. To empirically validate our theory, we design numerical tools to efficiently and economically examine the rank behavior of deep neural networks. This is a non-trivial task, as rank is very sensitive to noise and perturbation, and computing ranks of large networks is computationally prohibitive in time and space. Finally, we uncover an interesting phenomenon of independence deficit in multi-class classification networks. We find that many classes do not have their own unique representations in the classification network, and some highly irrelevant classes can decide the outputs of others. This independence deficit can significantly deteriorate the performance of networks in generalized data domains where each class demands a unique representation. In conclusion, the results of this work, together with the numerical tools we invent, may advance understanding of intrinsic properties of deep neural networks, and provide foundations for a broad study of low-dimensional structures in machine learning.

2 Preliminaries

Settings We consider the general deep neural network with L layers. It is a smooth vector-valued function $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^d$, where $\mathbb{R}^n$ and $\mathbb{R}^d$ are the ambient space of inputs and outputs, respectively. Deep neural networks are coupling of multiple layers, thus we write $\boldsymbol{F}$ as:

$$\boldsymbol{F} = \boldsymbol{f}^L \circ \boldsymbol{f}^{L-1} \circ \cdots \circ \boldsymbol{f}^1. \tag{1}$$

For simplicity, we further write the k-th sub-network[1] of $\boldsymbol{F}$ as

$$\boldsymbol{F}_k = \boldsymbol{f}^k \circ \cdots \circ \boldsymbol{f}^1, \tag{2}$$

and we use $\mathcal{F}_k = \boldsymbol{F}_k(\mathcal{X})$ to denote the feature space of the k-th sub-network on the data domain $\mathcal{X}$. We are more interested in the behavior of network rank in the feature spaces rather than scalar outputs (which trivially have rank 1). Thus for classification or regression networks that output a scalar value, we will consider $\boldsymbol{F} = \boldsymbol{F}_L$ as the transformation from the input space to the final feature space instead. Thus, we always have $n \gg 1$ and $d \gg 1$. For example, for ResNet-50 [21] architecture on ImageNet, we only consider the network slice from the inputs to the last feature layer of 2,048 units.

Rank of Function The rank of a function $\boldsymbol{f} = (\boldsymbol{f}_1, ..., \boldsymbol{f}_d)^T : \mathbb{R}^n \to \mathbb{R}^d$ refers to the highest rank of its Jacobian matrix $J_{\boldsymbol{f}}$ over its input domain $\mathcal{X}$, which is defined as

$$\mathrm{Rank}(\boldsymbol{f}) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}}) = \max_{\boldsymbol{x} \in \mathcal{X}} \mathrm{Rank}\left((\partial \boldsymbol{f}_i(\boldsymbol{x}) / \partial \boldsymbol{x}_j)_{n \times d} \right). \tag{3}$$

It is well-known that the region of non-highest rank is a zero-measure set on the feature manifold by Sard's theorem [23], so we are safe to ignore them in the definition of the function ranks. The rank of a function represents the volume of information captured by it in the output [23]. That is why it is so important to investigate the behavior of neural networks and many practical applications. Theoretically, by the rank theorem and Sard's theorem of manifolds [23], we can know that rank of the function equals the intrinsic dimension of its output feature space, as in the following lemma.[2]

Lemma 1. *Suppose that $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^d$ is smooth almost everywhere. Let $\mathrm{Rank}(\boldsymbol{f}) = r$. If data domain $\mathcal{X}$ is a manifold embedded in $\mathbb{R}^n$ and $\boldsymbol{\phi} : \mathcal{U} \to \mathcal{O}$ is a smooth bijective parameterization from an open subset $\mathcal{U} \subset \mathbb{R}^s$ to $\mathcal{O} \subset \mathcal{X}$, then we have $\dim(\boldsymbol{f}(\mathcal{X})) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f} \circ \boldsymbol{\phi}}) \leq r$. Thus, the rank of function $\boldsymbol{f}$ gives an upper bound for the intrinsic dimension $\dim(\boldsymbol{f}(\mathcal{X}))$ of the output space.*

It is worth mentioning that the intrinsic dimension $\dim(\boldsymbol{f}(\mathcal{X}))$ of the feature space is usually hard to measure, so the rank of the network gives an operational estimate of it.

[1] In this paper, sub-network means network slice from the input to some intermediate feature layer; layer network means an independent component of the network, without skip connections from the outside to it, like bottleneck layer of ResNet-50.

[2] Due to space limitation, all the related proofs are attached in the Appendix.

3 Numerical Tools

Validating the rank behavior of deep neural networks is a challenging task because it involves operations of high complexity on large-scale non-sparse matrices, which is infeasible both in time and space. Computing the full Jacobian representation of sub-networks of ResNet-50, for example, consumes over 150G GPU memory and several days at a single input point. In accuracy, this is even more challenging as rank is very sensitive to small perturbations. The numerical accuracy of $\mathrm{float}32$, $1.19e-7$ [43], cannot be trivially neglected in computing matrix ranks. Thus, in this section we establish some numerical tools for validating our subsequent arguments, and provide rigorous theoretical support for them.

3.1 Numerical Rank: Stable Alternative to Rank

The rank of large matrices is known to be unstable: it varies significantly under even small noise perturbations [46]. Matrices perturbed by even small Gaussian noises are almost surely of full rank, regardless of the true rank of the original matrix. Thus in practice we have to use an alternative: we count the number of singular values larger than some given threshold ϵ as the numerical rank of the matrix. Let $\boldsymbol{W} \in \mathbb{R}^{n\times d}$ be a given matrix. Its numerical rank with tolerance ϵ is

$$\mathrm{Rank}_\epsilon(\boldsymbol{W}) = \#\{i \in \mathbb{N}_+ : i \leq min\{n, d\}, \sigma_i \geq \epsilon\|\boldsymbol{W}\|_2\}, \tag{4}$$

where $\|\boldsymbol{W}\|_2$ is the ℓ_2 norm (spectral norm) of matrix W, $\sigma_i, i = 1, ..., \min\{n, d\}$ are its singular values, and $\#$ is the counting measurement for finite sets. We can prove that the numerical rank is stable under small perturbations. Based on Weyl inequalities [55], we have the following theorem.

Theorem 1. *For any given matrix $\boldsymbol{W}$, almost every tolerance $\epsilon > 0$, and any perturbation matrix $\boldsymbol{D}$, there exists a positive constant $\delta_{\max}(\epsilon)$ such that $\forall\delta \in [0, \delta_{\max}(\epsilon))$, $\mathrm{Rank}_\epsilon(\boldsymbol{W}+\delta\boldsymbol{D}) = \mathrm{Rank}_\epsilon(\boldsymbol{W})$. If $\boldsymbol{W}$ is a low-rank matrix without random perturbations, then there is a $\epsilon_{\max}$ such that for any $\epsilon < \epsilon_{\max}$, $\mathrm{Rank}_\epsilon(\boldsymbol{W} + \delta\boldsymbol{D}) = \mathrm{Rank}_\epsilon(\boldsymbol{W}) = \mathrm{Rank}(\boldsymbol{W})$ for all $\delta \in [0, \delta_{\max}(\epsilon))$.*

This property of the numerical rank metric makes it a suitable tool for investigating the rank behavior of neural networks. Possible small noises can be filtered out in Jacobian matrices of networks by using numerical rank. It is worth mentioning that random matrices no longer have full rank almost surely under the numerical rank. Instead their rank distribution can be inferred from the well-known Marcenko–Pastur distribution [36] of random matrices. So under numerical rank, low-rank matrices will be commonly seen. In this paper, we always use the numerical rank when measuring ranks.

3.2 Partial Rank of the Jacobian: Estimating Lower Bound of Lost Rank in Deep Networks

To enable the validation of trend of the network ranks, we propose to compute only the rank of sub-matrices of the Jacobian as an alternative. Those sub-matrices are also the Jacobian matrices with respect to a fixed small patch of inputs. Rigorously, given a function $\boldsymbol{f}$ and its Jacobian $\boldsymbol{J_f}$, we denote partial rank of the Jacobian as the rank of a sub-matrix of the Jacobian that consists of the j_1-th, j_2-th,...,j_K-th column of the original Jacobian

$$\mathrm{PartialRank}(\boldsymbol{J_f}) = \mathrm{Rank}(\mathrm{Sub}(\boldsymbol{J_f}, j_1, ..., j_K)) = \mathrm{Rank}((\partial\boldsymbol{f}_i/\partial\boldsymbol{x}_{j_k})_{d\times K}), \tag{5}$$

where $1 \leq j_1 < \ldots < j_K \leq n$. The partial rank can be efficiently computed and the variance of partial ranks of adjacent sub-networks gives a lower bound on the variance of their ranks. We demonstrate this in appendix F.

3.3 Classification Dimension: Estimating Final Feature Dimension

Measuring the intrinsic dimension of feature manifolds is known to be intractable. So we turn to an approximation procedure. For most classification networks, a linear regression over the final feature manifold decides the final network prediction and accuracy. So we can estimate the intrinsic dimension as the minimum number of principal components in the final feature space to preserve a high classification accuracy. Let $\mathrm{cls} : \mathbb{R}^d \to \mathbb{R}^c$ be the classification predictions based on the final

Networks	ResNet-18	ResNet-50	GluMixer-24	ResMLP-S24	Swin-T	ViT-T
ClsDim	149	131	199	196	344	109
Ambient Dim.	512	2048	384	384	768	192

Table 1: Classification dimensions (with respect to 95% classification performance of the ambient feature space $\mathbb{R}^d$) and ambient dimensions of the final feature manifolds of different networks. All networks have low intrinsic dimensions for final features.

feature representation $\boldsymbol{F}(\boldsymbol{x})$, Pro_k be the operator that project a vector to the subspace spanned by the top-k PCA components of the final feature representations, and $\mathbf{1}_{\mathrm{cond}}$ the indicator for condition cond. The classification dimension is then defined as

$$\mathrm{ClsDim}(\boldsymbol{F}(\mathcal{X})) = \min_k \{k : \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{P}_{\mathcal{X},\mathcal{Y}}}[\mathbf{1}_{\mathrm{Cls}(\mathrm{Pro}_k(\boldsymbol{F}(\boldsymbol{x})))==\boldsymbol{y}}] \geq 1-\epsilon\}, \tag{6}$$

which is the minimum dimensionality needed to reconstruct the classification accuracy of the whole model.

4 Simple Principle of Rank Diminishing for General Networks

The specific designs of neural networks are vast and diverse, but most of them share two fundamental ingredients of deep networks, *i.e.*, *the chain rule and matrix multiplication*. This provides us a chance to analyze the behavior of deep networks that is architecture-agnostic. So in the theoretical aspect, we will focus on how these two intrinsic structures endow impetus of rank diminishing to neural networks.

A most straightforward observation comes from the basic rule of matrix multiplication that for any two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we have $\mathrm{Rank}(\boldsymbol{AB}) \leq \min\{\mathrm{Rank}(\boldsymbol{A}), \mathrm{Rank}(\boldsymbol{B})\}$ [24]. Taking this into the chain rule of differential of $\boldsymbol{J}_{\boldsymbol{F}} = \boldsymbol{J}_{\boldsymbol{f}^L}\boldsymbol{J}_{\boldsymbol{f}^{L-1}}...\boldsymbol{J}_{\boldsymbol{f}^1}$, we then have $\mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{F}_k}) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}^k\circ\boldsymbol{F}_{k-1}}) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}^k}\boldsymbol{J}_{\boldsymbol{F}_{k-1}}) \leq \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{F}_{k-1}}), k = 2, ..., L$, which is Eq. (7). We can then get the following principle of rank diminishing.

Theorem 2 (Principle of Rank Diminishing)**.** *Suppose that each layer $\boldsymbol{f}_i, i = 1, ..., L$ of network $\boldsymbol{F}$ is almost everywhere smooth*[3] *and data domain $\mathcal{X}$ is a manifold, then both the rank of sub-networks and intrinsic dimension of feature manifolds decrease monotonically by depth:*

$$\mathrm{Rank}(\boldsymbol{f}_1) \geq \mathrm{Rank}(\boldsymbol{f}_2 \circ \boldsymbol{f}_1) \geq ... \geq \mathrm{Rank}(\boldsymbol{f}_{L-1} \circ ... \circ \boldsymbol{f}_1) \geq \mathrm{Rank}(\boldsymbol{F}_L), \tag{7}$$

$$\dim(\mathcal{X}) \geq \dim(\mathcal{F}_1) \geq \dim(\mathcal{F}_2) \geq ... \geq \dim(\mathcal{F}_L). \tag{8}$$

This principle describes the behavior of generic neural networks with almost everywhere smooth components, which exhibits the monotonic decreasing (but not strictly) of network ranks and intrinsic dimensionality of feature manifolds.

A flaw of the above principle is that it does not tell whether the rank must decrease. So we need further analysis for the chance of strictly decreasing. A most direct reason to support the strictly decreasing comes from the structural impetus of rank deficiency of numerous network components. Frequently used operations like pooling, downsampling, and dense layer can loose ranks considerably as they explicitly decrease the ambient dimensions of feature representations, or have low rank weight matrices [37]. Specifically, we can have a global criterion for whether a network component will lose ranks as follows.

Theorem 3 (Structural Impetus of Strictly Decreasing)**.** [4] *Roughly speaking, if almost everywhere on the input feature manifold, there is a direction such that moving along this direction keeps the output invariant, then the intrinsic dimension of the output feature manifold will be strictly lower than that of the input. The maximum number of independent such directions gives a lower bound on the number of lost intrinsic dimensions.*

[3] It means having arbitrary order gradients except for a zero measure set in the input domain

[4] The rigorous version is given in the Appendix.

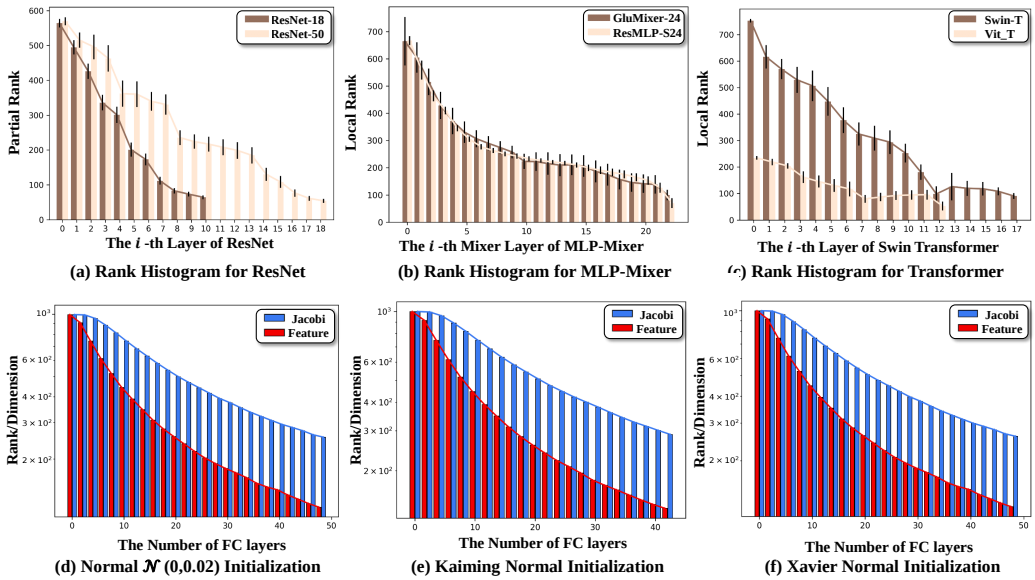


Figure 1: Partial rank of Jacobian matrices of CNN, MLP, and Transformer architecture networks for different layers on ImageNet (top row); rank of Jacobian matrices and feature dimensions of linear MLP network following conditions of Theorem 5 (bottom row). All the models show a similar trend of fast decreasing of ranks as predicted by Theorems 4 and 5.

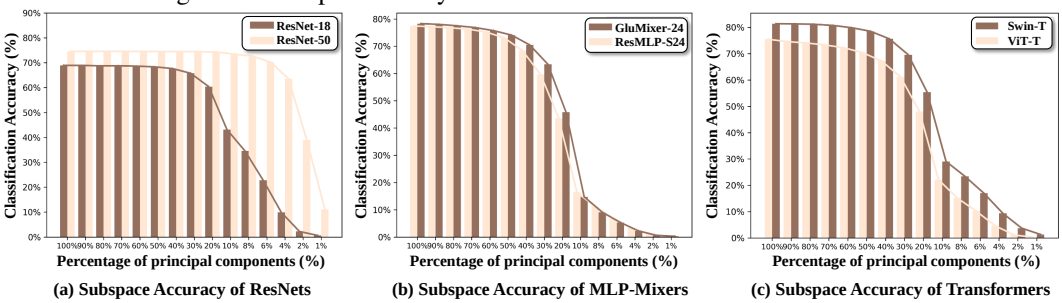


Figure 2: Classification Accuracy (top-1) of using subspaces spanned by top-$k\%$ eigenvectors (principal components) of the final feature manifolds. For all networks a small percentage (see Tab. 1) of eigenvectors are enough to reproduce the classification accuracy of the whole network, indicating a low intrinsic dimension of final feature manifolds. **Note that the x-axes are non-linear.**

For example, the ReLU activation gives the same output for all inputs that only differs from each other in the negative parts. Thus the feature manifold after a ReLu activation will lose the dimension for those negative parts. For general linear layers (*e.g.,* pure convolution and dense layers), the feature dimensions that belong to the orthogonal complementary spaces of the weights will be lost after applying the linear transformations, as the weight matrices are inactive to changes in their orthogonal complementary spaces.

5 Limiting Behavior of Network Ranks for Infinitely Deep Networks

From a theoretical perspective, it is of great interest to consider the limiting behavior—the rank of infinitely deep neural networks. However, it is infeasible to directly consider this problem and give any rigorous analysis for general cases— it is impossible to exhaust the tedious discussion of how specific structures can influence network ranks. Thus we need to concentrate to a simplified but still representative math model of this problem.

Theory Setup and Its Necessity In brief, we will assume in this section that the Jacobi matrices are random matrices independently following some distributions in a fixed Euclidean space. We then consider the limiting behavior of the singular values of series $\{\boldsymbol{J}_{\boldsymbol{F}_n}\}_{n=1}^{\infty}$. Modeling the Jacobi

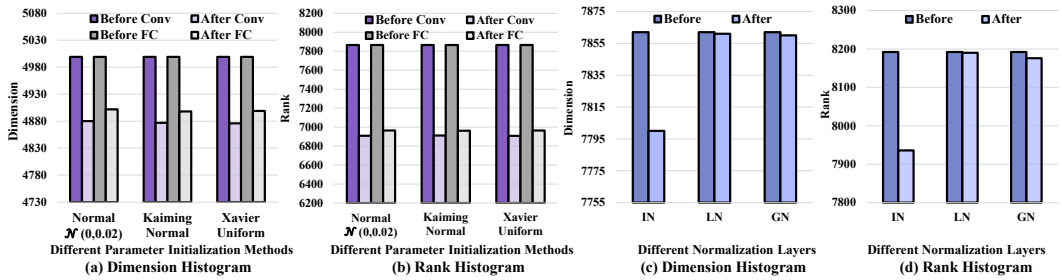


Figure 3: PCA dimension of feature spaces and rank of Jacobian matrix for commonly seen network components under standard Gaussian inputs and randomized weights. Convolution and FC layers tend to lose rank considerably; normalization layers, like InstanceNorm (IN) [52], LayerNorm (LN) [2], and GroupNorm (GN) [58], lose rank modestly. But none can preserve rank.

matrices with random matrices allows us to neglect the influence of specific architectures and reveal the intrinsic property using large number law of probability. While limiting them in the same space is a common practice in analyzing infinitely deep networks [48, 40, 44, 3, 34], as it offers the feasibility to define limits of variables. For example, if the sizes of Jacobi matrices changes from time to time, some singular values may suddenly disappear or appear in the series $\{\boldsymbol{J}_{\boldsymbol{F}_n}\}_{n=1}^{\infty}$, thus it is unable to define limits for them.

After this simplification, the problem is still representative for understanding what happens in reality. The transformer, recurrent neural networks, and some very deep CNN architectures with constant width [59] can be viewed as examples of this setting. Another interesting setting is to assume that the Jacobi matrices follow the most simple distribution—Gaussian distributions. This can lead to a much stronger yet intuitive result on network ranks. So we are most interested in the limiting behavior when the Jacobi matrices follow weak regularized distributions and the Gaussian distributions. These two cases then lead to Theorems 4 and 5 correspondingly.

Theorem 4. *Let the network be $\boldsymbol{F} = \boldsymbol{f}^L \circ \cdots \circ \boldsymbol{f}^1$, and all the ambient dimensions of feature manifolds be the same as the ambient dimension of inputs, i.e., $\boldsymbol{f}^k : \mathbb{R}^n \to \mathbb{R}^n, k = 1, \ldots, L$. Suppose the Jacobian matrix of each layer $\boldsymbol{f}_i$ independently follows some distribution μ, and $\mathbb{E}_\mu[\max\{\log \|\boldsymbol{J}_{\boldsymbol{f}^k}^{\pm 1}\|_2, 0\}] < \infty$. Let σ_k denote the k-th largest singular value of $\boldsymbol{J}_{\boldsymbol{F}}$. Then there is an integer $r < n$ and positive constants $\mu_r, \ldots, \mu_n$ that only depend on μ such that we have for μ-almost everywhere,*

$$\frac{\sigma_k}{\|\boldsymbol{J}_{\boldsymbol{F}}\|_2} \sim \exp(-L\mu_k) \to 0, k = r, \ldots, n, \text{ as } L \to \infty, \tag{9}$$

meaning that for any tolerance $\epsilon > 0$, $\mathrm{Rank}_\epsilon(\boldsymbol{F})$ drops below $r + 1$ as $L \to \infty$.

Theorem 5. *Let the network be $\boldsymbol{F} = \boldsymbol{f}^L \circ ... \circ \boldsymbol{f}^1$, and all the ambient dimensions of feature manifolds be the same as the ambient dimension of inputs, i.e., $\boldsymbol{f}^k : \mathbb{R}^n \to \mathbb{R}^n, k = 1, ..., L$. Suppose that $\boldsymbol{J}_{\boldsymbol{f}^i}$ independently follows the standard Gaussian distribution. Let σ_k denote the k-th largest singular value of $\boldsymbol{J}_{\boldsymbol{F}}$. Then almost surely*

$$\lim_{L\to\infty} \left(\frac{\sigma_k}{\|\boldsymbol{J}_{\boldsymbol{F}}\|_2}\right)^{\frac{1}{L}} = \exp\frac{1}{2}\left(\psi(\frac{n-k+1}{2}) - \psi(\frac{n}{2})\right) < 1, k = 2, \ldots, n, \tag{10}$$

where $\psi = \Gamma/\Gamma'$ and Γ is the Gamma function. That means for a large L and any tolerance ϵ, $\mathrm{Rank}_\epsilon(\boldsymbol{F})$ drops to 1 exponentially with speed nC^L, where $C < 1$ is a positive constant that only depends on n.

The proofs of these two theorems rely on the advances of random matrix theory, especially the study of Lyapunov exponents of linear co-cycle systems [49, 57, 17, 45, 31] (see appendix A for detail). Both of the two theorems reveal that infinitely deep networks have an intrinsic intention to drop ranks rapidly and centralize energy in the largest singular vectors. This intention is purely aroused by the two essential ingredients, chain rule and matrix multiplication, as here we omit the consideration of specific network designs, but focus on the large number law of coupling of massive smooth functions.

These two theorems point out that the ratio of non-largest singular value divided by the largest one will converge to zero with an exponential speed. As a result, the numerical rank of the network will decrease quickly after the layer is deep enough. While this is not the hard rank we use in Theorem 2, we can still use it to infer the low rank structure of feature representations. Let $\boldsymbol{q}_1, \cdots, \boldsymbol{q}_n$ be the singular vectors of $\boldsymbol{J}_{\boldsymbol{F}}$ corresponding to the singular values in Theorem 4, and $\alpha_i = \langle \Delta \boldsymbol{x} / \|\Delta \boldsymbol{x}\|_2, \boldsymbol{q}_i \rangle$, then when the depth $L \to \infty$ and $\Delta \boldsymbol{x}$ is small, we have $\boldsymbol{F}(\boldsymbol{x}+\Delta \boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{J}_{\boldsymbol{F}} \Delta \boldsymbol{x} + o(\|\Delta \boldsymbol{x}\|_2) = \sigma_1 \sum_{i=1}^{r} \alpha_i \boldsymbol{q}_i + \sigma_1 \sum_{j=2}^{n} \frac{\sigma_j}{\sigma_1} \alpha_j \boldsymbol{q}_j + o(\|\Delta \boldsymbol{x}\|_2) \approx \sigma_1 \sum_{i=1}^{r} \alpha_i \boldsymbol{q}_i$, which means the neighborhood of data $\boldsymbol{x}$ is mapped into an r-dimensional subspace spanned by the singular vectors of the largest singular values.

This can also explain why layer-wise regularization techniques are helpful for training deep networks, as those regularization techniques re-normalize the singular value distributions of deep networks, easing the trend of the ratio $\frac{\sigma_i}{\sigma_1}$ becoming infinitesimal. Specifically, if the data distribution near $\boldsymbol{x}$ is assumed to be standard Gaussian, then the feature distribution is $\mathcal{N}(\boldsymbol{F}(\boldsymbol{x}), \boldsymbol{J}_{\boldsymbol{F}}^T \boldsymbol{J}_{\boldsymbol{F}})$, which has a very low rank covariance matrix due to $\frac{\sigma_i}{\sigma_1}, i \geq r$ is tiny. Then applying Batch Normalization will try to pull this feature distribution back to Gaussian with identity covariance, thus ease the trend of feature dimension collapse. This has also been discussed in related work of the Batch Normalization [13, 5]. We will discuss some other techniques to remiss the rank diminishing in the Appendix.

These two theorems are also connected with the famous gradient explosion issue of deep neural networks [4, 42], where the largest singular value of the Jacobian matrix tends to infinity when the layer gets deeper. This issue could be viewed as a special case of Theorem 5 that investigates the behavior of all singular values of deep neural networks. The behavior of network ranks in fact manipulates the well-known gradient explosion issue. Rigorously, we have the following conclusion.

Corollary 1. *Under the condition of Theorem 5, then almost surely gradient explosion happens at an exponential speed, i.e.,* $\log \|\boldsymbol{J}_{\boldsymbol{F}}\|_2 = \log \sigma_1 \sim \frac{L}{2}(\log 2 + \psi(n/2)) \to \infty$ *when* L *is large.*

6 Validating the Rank Behavior

6.1 Validating Rank Diminishing by Depth

In this section, we numerically validate our theory in three types of architectures of benchmark deep neural networks, CNNs, MLPs, and Transformers, in the ImageNet [14] data domain. Information of those networks is listed in Tab. A2. For validating the tendency of network rank of Jacobian matrices, we use the numerical rank of sub-matrices of Jacobian on the central $16 \times 16 \times 3$ image patch of input images. We report the results of other choices of patches in the Appendix. Details of the experiment setup can be found in appendix E.

Rank Diminishing of Jacobians As is discussed in Sec. 3.2 and Lemma 2, the partial rank of the Jacobian is a powerful weapon for us to detect the behavior of huge Jacobian matrices, which are infeasible to compute in practice. The decent value of partial ranks of adjacent sub-networks provides a lower bound to that of full ranks of them. Fig. 1 (a,b,c) report the partial rank of Jacobian matrices of three types of architectures, where we can find consistent diminishing of partial ranks in each layer, indicating a larger rank losing for the full rank of Jacobian matrices.

Implicit Impetus Theorem 5 gives an exponential speed of rank decent by layers. We find that it corresponds well with practice. We investigate this exponential law in a toy network of MLP-50, which is composed of 50 dense layers, each with 1,000 hidden units. The MLP-50 network takes Gaussian noise vectors of $\mathbb{R}^{1000}$ as inputs, and returns a prediction of 1,000 categories. As all the feature manifolds are linear subspaces in this case, their intrinsic dimensions can be directly measured by the numerical rank of their covariance matrices. We report the full rank of Jacobian matrices and intrinsic dimensions of feature manifolds under three different randomly chosen weights in Fig. 1 (d,e,f). Due to the digital accuracy of $\mathrm{float32}$, we stop calculation in each setting when the absolute values of elements of the matrices are lower than $1.19e-7$. We can find standard curves of exponential laws in all cases for both ranks of Jacobian and intrinsic dimensions of features.

Structural Impetus We validate the structural impetus in Fig. 3. To give an estimation for general cases, here we use Gaussian noises with the size of $128 \times 8 \times 8$ as inputs, and randomize weights of the network components to be validated. We plug those components into a simple fully-connected (FC) layer of 8,192 hidden units. As the structure is simple, we directly measure the intrinsic dimension of feature spaces and the full rank of Jacobian matrices before and after the features pass the network components to be measured. The dimension is determined by the number of PCA eigenvalues [25, 56] larger than $1.19e-7 \times N \times \sigma_{\max}$, where N is the number of PCA eigenvalues, and $\sigma_{\max}$ is the largest PCA eigenvalue. The batch size is set to 5,000. We find that the convolution (the kernel size is 3×3) and FC layers (the weight size is 8,192) tend to lose rank considerably, while different normalization layers also lose rank modestly. But none of them can preserve rank invariant.

6.2 Validating Low-Rank Terminal Spaces

This section seeks evidence to support a low-rank terminal feature space of deep networks, which can then support the significant diminishing of ranks in the previous layers. The evidence is consisted of a numerical validation that investigates the classification dimension of the terminal feature layer, together with a semantic validation that reveals the independence deficit of deep networks.

Numerical Validation To get an estimation of how many dimensions remain in the final feature representation, we measure the classification dimension in Fig. 2 and Tab. 1. We report the classification accuracy produced by projecting final feature representations to its top $k\%$ eigenvectors in Fig. 2. We choose a threshold of ϵ such that this procedure can reproduce 95% of the original accuracy of the network. The corresponding ClsDim is reported in Tab. 1. As discussed in Sec. 3.3, this gives an estimation of the intrinsic dimension of the final feature manifold. We can find a universal low-rank structure for all types of networks.

Semantic Validation We want to show that there are only a few independent representations to decide the classification scores for all the 1,000 categories of ImageNet. Specifically, can we predict the outputs of the network for some categories based on the outputs for a few other categories, as illustrated in Fig. 4 (a)? And if we can, will those categories be strongly connected to each other? A surprising fact is that, we can find many counter examples of irrelevant categories dominating the network outputs for given categories regarding various network architectures. This interesting phenomenon indicates a rather drastic competing in the final feature layer for the tight rank budgets of all categories, which yields non-realistic dependencies of different categories.

To find the dependencies of categories in final features, we can solve the following Lasso problem [50],

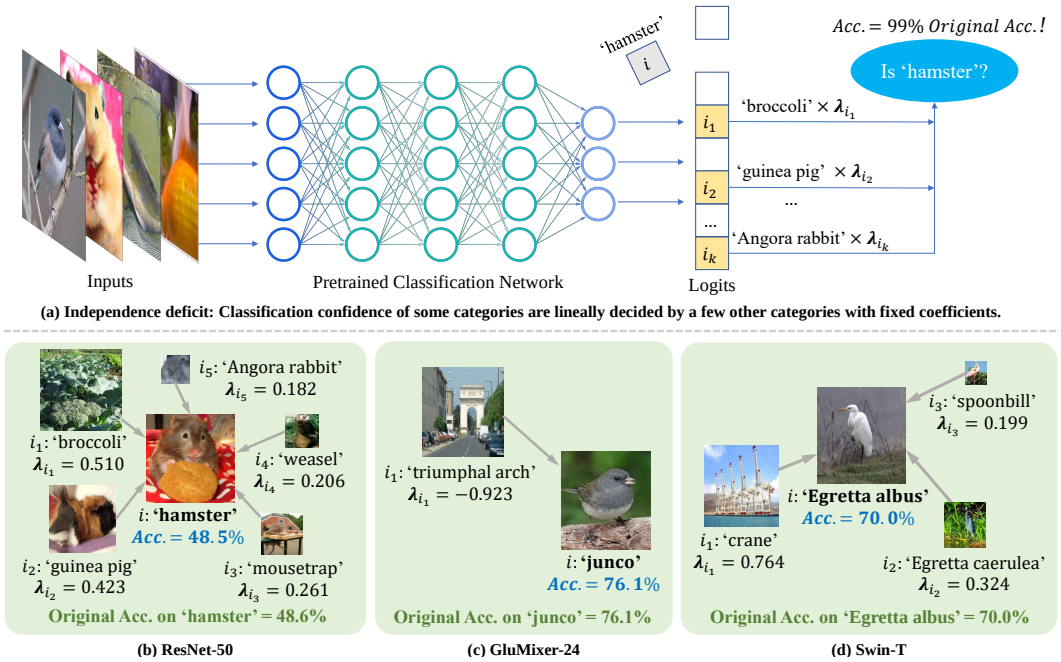
$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}_i = -1} \mathbb{E}_{\boldsymbol{x}}[\|\boldsymbol{\lambda}^T \boldsymbol{W} \boldsymbol{F}(\boldsymbol{x})\|_2^2] + \eta \|\boldsymbol{\lambda}\|_1, \tag{11}$$

where $\boldsymbol{F}(\boldsymbol{x}) \in \mathbb{R}^{1000}$ is the slice of network from inputs to the final feature representation, $\boldsymbol{x}$ is the sample from ImageNet $\mathcal{X}$, and $\boldsymbol{W}$ is the final dense layer. The solution $\boldsymbol{\lambda}^*$ will be a sparse vector, with k non-zero elements $\lambda_{i_1} \geq \lambda_{i_2} \geq ... \geq \lambda_{i_k}, k \ll 1000$. We can then get

$$\mathrm{logits}(\boldsymbol{x}, i) \approx \boldsymbol{\lambda}_{i_1} \mathrm{logits}(\boldsymbol{x}, i_1) + ... + \boldsymbol{\lambda}_{i_k} \mathrm{logits}(\boldsymbol{x}, i_k), i \notin \{i_1, ..., i_k\}, k \ll 1000, \forall \boldsymbol{x} \in \mathcal{X}, \tag{12}$$

where $\mathrm{logits}(\boldsymbol{x}, i_j), j = 1, \ldots, k$ is the logits of network for category i_j, *i.e.*, $\mathrm{logits}(\boldsymbol{x}, i_j) = \boldsymbol{W}_{i_j} \boldsymbol{F}(\boldsymbol{x})$. It is easy to see that outputs for category i are linearly decided by outputs for $i_1, ..., i_k$ and are dominated by outputs for i_1.

Fig. 4 demonstrates the solutions of Eq. (12) for three different categories in ImageNet with $\eta = 20$, and network architectures ResNet-50, GluMixer-24, and Swin-T. The results are surprising. It shows that many categories of the network predictions are in fact 'redundant', as they are purely decided by the predictions of the other categories with simple linear coefficients. In this case, the entanglement of different categories cannot be avoided, thus the network may perform poorly under domain shift. An even more surprising finding is that, some very irrelevant categories hold the largest weights when deciding the predictions of the redundant categories, which means that the networks just neglect the unique representations of those categories in training and yield over-fitting when predicting them.



(a) Independence deficit: Classification confidence of some categories are lineally decided by a few other categories with fixed coefficients.

(b) ResNet-50 **(c) GluMixer-24** **(d) Swin-T**

Figure 4: Independence deficit. Classification confidence of some ImageNet categories are lineally decided by a few other categories with fixed coefficients in the whole data domain. We illustrate this phenomenon in (a). Here we present some results from ResNet-50, GluMixer-24, and Swin-T. In the (b,c,d) we illustrate the categories of $i_1, ..., i_k$ (in the surrounding) to linearly decide category i (in the center) and their corresponding weights $\lambda_{i_1}, ..., \lambda_{i_k}$. The classification accuracy on the validation set of using Eq. (12), instead of the true logits, to predict the label is reported in **blue** (if tested on positive samples only, the accuracy rates are 98%, 90%, 82% for cases in (b,c,d) correspondingly). For comparison, the original accuracy for the corresponding categories are reported in **green**. We can find that 1) a few other categories can decide the confidence of the target category i; 2) some very irrelevant categories contribute the largest weights. For example in (c), the logits of class 'junco' is the negative of 'triumphal arch'. Both of them indicate a rather drastic competition of different categories for independent representations in final features due to the tight rank budgets.

7 Related Work

Previous studies of rank deficiency in deep neural networks follow two parallel clues. One is the study of rank behavior in specific neural network architectures. [15] studies deep networks consisting of pure self-attention networks, and proves that they converge exponentially to a rank-1 matrix under the assumption of globally bounded weight matrices. [13] studies the effect of BatchNorm on MLPs and shows that BatchNorm can prevent drastic diminishing of network ranks in some small networks and datasets. Both of those works avoid directly validating the behavior of network ranks in intermediate layers due to the lacking of efficient numerical tools. An independent clue is the study of implicit self-regularization, which finds that weight matrices tend to lose ranks after training. [37] studies this phenomenon in infinitely-wide, over-parametric neural networks with tools from random matrix theory. [1] studies this phenomenon in deep matrix decomposition. Those works focus on the theoretical behavior of rank of weight matrices induced by the training instead of network ranks.

8 Conclusion

This paper studies the rank behavior of deep neural networks. In contrast to previous work, we focus on directly validating rank behavior with deep neural networks of diverse benchmarks and various settings for real scenarios. We first formalize the analysis and measurement of network ranks. Then under the proposed numerical tools and theoretical analysis, we demonstrate the universal rank diminishing of deep neural networks from both empirical and theoretical perspectives. We further support the rank-deficient structure of networks by revealing the independence deficit phenomenon, where network predictions for a category can be linearly decided by a few other, even irrelevant categories. The results of this work may advance understanding of the behavior of fundamental network architectures and provide intuition for a wide range of work pertaining to network ranks.

References

[1] S. Arora, N. Cohen, W. Hu, and Y. Luo. Implicit regularization in deep matrix factorization. *Adv. Neural Inform. Process. Syst.*, 32, 2019.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *Adv. Neural Inform. Process. Syst.*, 2019.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5(2):157–166, 1994.

[5] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger. Understanding batch normalization. *Adv. Neural Inform. Process. Syst.*, 31, 2018.

[6] C. Blakeney, Y. Yan, and Z. Zong. Is pruning compression?: Investigating pruning via network layer similarity. In *IEEE Winter Conf. Appl. Comput. Vis.*, pages 914–922, 2020.

[7] X. Cai, C. Ding, F. Nie, and H. Huang. On the equivalent of low-rank linear regressions and linear discriminant analysis based regressions. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 1124–1132, 2013.

[8] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré. Scatterbrain: Unifying sparse and low-rank attention. *Adv. Neural Inform. Process. Syst.*, 34, 2021.

[9] P. Chen, S. Si, Y. Li, C. Chelba, and C.-J. Hsieh. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. *Adv. Neural Inform. Process. Syst.*, 31, 2018.

[10] P. Chen, H.-F. Yu, I. Dhillon, and C.-J. Hsieh. Drone: Data-aware low-rank compression for large NLP models. *Adv. Neural Inform. Process. Syst.*, 34:29321–29334, 2021.

[11] Y. Cheng, L. Yin, and Y. Yu. Lorslim: Low rank sparse linear methods for top-n recommendations. In *IEEE Int. Conf. on Data Min.*, pages 90–99. IEEE, 2014.

[12] J. Chiu, Y. Deng, and A. Rush. Low-rank constraints for fast inference in structured models. *Adv. Neural Inform. Process. Syst.*, 34, 2021.

[13] H. Daneshmand, J. Kohler, F. Bach, T. Hofmann, and A. Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Adv. Neural Inform. Process. Syst.*, 33:18387–18398, 2020.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255. Ieee, 2009.

[15] Y. Dong, J.-B. Cordonnier, and A. Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *Int. Conf. Mach. Learn.*, pages 2793–2803. PMLR, 2021.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2020.

[17] H. Furstenberg and H. Kesten. Products of random matrices. *Ann. Math. Stat.*, 31(2):457–469, 1960.

[18] H. Furstenberg and H. Kesten. Random matrix products and measures on projective spaces. *Israel J. Math.*, 46:12–32, 1983.

[19] D. Goldfarb and Z. Qin. Robust low-rank tensor recovery: Models and algorithms. *SIAM J. Matrix Anal. Appl.*, 35(1):225–253, 2014.

[20] B. Hanin and D. Rolnick. How to start training: The effect of initialization and architecture. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Adv. Neural Inform. Process. Syst.*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/d81f9c1be2e08964bf9f24b15f0e4900-Paper.pdf.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[22] D. Hendrycks and K. Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[23] M. W. Hirsch. *Differential topology*, volume 33. Springer Science & Business Media, 2012.

[24] K. Hoffman. *Linear algebra*. Englewood Cliffs, NJ, Prentice-Hall, 1971.

[25] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[26] C.-J. Hsieh, K.-Y. Chiang, and I. S. Dhillon. Low rank modeling of signed networks. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 507–515, 2012.

[27] Y.-C. Hsu, T. Hua, S. Chang, Q. Lou, Y. Shen, and H. Jin. Language model compression with weighted low-rank factorization. In *Int. Conf. Learn. Represent.*, 2021.

[28] L. Jing, L. Yang, J. Yu, and M. K. Ng. Semi-supervised low-rank mapping learning for multi-label classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1483–1491, 2015.

[29] R. Karimi Mahabadi, J. Henderson, and S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Adv. Neural Inform. Process. Syst.*, 34, 2021.

[30] M. Kheirandishfard, F. Zohrizadeh, and F. Kamangar. Deep low-rank subspace clustering. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, pages 864–865, 2020.

[31] J. F. C. Kingman. Subadditive ergodic theory. *Ann. Probab.*, pages 883–899, 1973.

[32] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 365–374, 2017.

[33] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao. Hrank: Filter pruning using high-rank feature map. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1529–1538, 2020.

[34] J. Liu, K. Kawaguchi, B. Hooi, Y. Wang, and X. Xiao. Eignn: Efficient infinite-depth graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Adv. Neural Inform. Process. Syst.*, volume 34, pages 18762–18773. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/9bd5ee6fe55aaeb673025dbcb8f939c1-Paper.pdf.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10012–10022, 2021.

[36] V. A. Marčenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Math. USSR Sb.*, 1(4):457, 1967.

[37] C. H. Martin and M. W. Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *J. Mach. Learn. Res.*, 22(165):1–73, 2021.

[38] J. Mu, R. Xiong, X. Fan, D. Liu, F. Wu, and W. Gao. Graph-based non-convex low-rank regularization for image compression artifact reduction. *IEEE Trans. Image Process.*, 29:5374–5385, 2020.

[39] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Int. Conf. Mach. Learn.*, pages 807–814. PMLR, 2010.

[40] A. Nazaret and D. Blei. Variational inference for infinitely deep neural networks. In *Int. Conf. Mach. Learn.*, pages 16447–16461. PMLR, 2022.

[41] C. M. Newman. The distribution of Lyapunov exponents: Exact results for random matrices. *Commun. Math. Phys.*, 103(1):121–126, 1986.

[42] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Int. Conf. Mach. Learn.*, pages 1310–1318. PMLR, 2013.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inform. Process. Syst.*, 32, 2019.

[44] S. Peluchetti and S. Favaro. Doubly infinite residual neural networks: a diffusion process approach. *J. Mach. Learn. Res.*, 22(175):1–48, 2021. URL http://jmlr.org/papers/v22/20-706.html.

[45] Y. B. Pesin. Characteristic Lyapunov exponents and smooth ergodic theory. *Russ. Math. Surv.*, 32(4):55, 1977.

[46] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*, volume 37. Springer Science & Business Media, 2010.

[47] N. Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[48] S. Sonoda and N. Murata. Transport analysis of infinitely deep neural network. *J. Mach. Learn. Res.*, 20 (1):31–82, 2019.

[49] R. Temam. *Infinite-dimensional dynamical systems in mechanics and physics*, volume 68. Springer Science & Business Media, 2012.

[50] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B Stat. Methodol.*, 58(1):267–288, 1996.

[51] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, et al. ResMLP: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021.

[52] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[53] T. Vogels, S. P. Karimireddy, and M. Jaggi. Practical low-rank communication compression in decentralized deep learning. *Adv. Neural Inform. Process. Syst.*, 33:14171–14181, 2020.

[54] H. Wang and N. Ahuja. Rank-r approximation of tensors using image-as-matrix representation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 2, pages 346–353. IEEE, 2005.

[55] H. Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen. *Math. Ann.*, 71:441–479, 1912.

[56] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemom. Intell. Lab. Syst*, 2(1-3): 37–52, 1987.

[57] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining Lyapunov exponents from a time series. *Physica D*, 16(3):285–317, 1985.

[58] Y. Wu and K. He. Group normalization. In *Eur. Conf. Comput. Vis.*, pages 3–19, 2018.

[59] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In J. Dy and A. Krause, editors, *Int. Conf. Mach. Learn.*, volume 80 of *Proceedings of Machine Learning Research*, pages 5393–5402. PMLR, 10–15 Jul 2018.

[60] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *Int. Conf. Mach. Learn.*, pages 5393–5402. PMLR, 2018.

[61] J. Ye. Generalized low rank approximations of matrices. *Mach. Learn.*, 61(1):167–191, 2005.

[62] X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7370–7379, 2017.

[63] Z. Zha, X. Yuan, B. Wen, J. Zhou, J. Zhang, and C. Zhu. From rank estimation to rank approximation: Rank residual constraint for image restoration. *IEEE Trans. Image Process.*, 29:3254–3269, 2019.

[64] M. Zhan, S. Cao, B. Qian, S. Chang, and J. Wei. Low-rank sparse feature selection for patient similarity learning. In *IEEE Int. Conf. on Data Min.*, pages 1335–1340. IEEE, 2016.

[65] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja. Low-rank sparse coding for image classification. In *Int. Conf. Comput. Vis.*, pages 281–288, 2013.

[66] Y. Zhang, Z. Jiang, and L. S. Davis. Learning structured low-rank representations for image classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 676–683, 2013.

[67] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *J. Comput. Graph. Stat.*, 15(2): 265–286, 2006.

Appendix

Contents

A Proofs **13**

A.1 Proof to Lemma 1 13

A.2 Proof to Theorem 1 14

A.3 Proof to Lemma 2 15

A.4 Proof to Theorem 2 15

A.5 Proof to Theorem 3 15

A.6 Proof to Theorem 4 16

A.6.1 Lyapunov exponents are limits of logarithms of subspace spectral norm divided by layer depth L 17

A.6.2 Singular value distributions of Jacobian matrices of deep function coupling 18

A.6.3 Existence of Lyapunov exponents for Jacobian matrices of deep function coupling 20

A.7 Proof to Theorem 5 20

A.8 Proof to Corollary 1 20

B Influences of Structures **20**

C Orthogonal Initialization **22**

D Code **22**

E Experiments Setup **23**

F Partial Rank of the Jacobian: Estimating Lower Bound of Lost Rank in Deep Networks **23**

F.1 Partial Rank of Jacobians under Different Input Patches 23

G Estimating Dimension Diminishing in Features **24**

H More Examples of Independence Deficit **25**

A Proofs

A.1 Proof to Lemma 1

This Lemma is the direct result of the *rank theorem of manifolds*.

Theorem 6 (Rank Theorem [23])**.** *Suppose $\boldsymbol{f} : \mathcal{M} \to \mathcal{N}$ is a smooth function from m-dimensional manifold $\mathcal{M}$ to n-dimensional manifold $\mathcal{N}$, and $\mathrm{Rank}_{\mathcal{M},\mathcal{N}}(\boldsymbol{J}_{\boldsymbol{f}}) = r$. Then for each $\boldsymbol{x} \in \mathcal{M}$, there exists a smooth chart $(\mathcal{U}, \boldsymbol{m})$ around $\boldsymbol{x}$ and a smooth chart $(\mathcal{V}, \boldsymbol{n})$ around $\boldsymbol{f}(\boldsymbol{x})$, such that*

$$\boldsymbol{n} \circ \boldsymbol{f} \circ \boldsymbol{m}^{-1} : \boldsymbol{m}(\mathcal{U}) \subset \mathbb{R}^m \to \boldsymbol{n}(\mathcal{V}) \subset \mathbb{R}^n \tag{A13}$$

is given by $\boldsymbol{n} \circ \boldsymbol{f} \circ \boldsymbol{m}^{-1}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m) = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m, 0, \cdots, 0)$.

The rank of a function $\mathrm{Rank}_{\mathcal{M},\mathcal{N}}$ defined on the manifold is the rank under local chart systems of the input manifold $\mathcal{M}$ and output manifold $\mathcal{N}$. Let $\boldsymbol{\phi}$ be the chart for point $\boldsymbol{x} \in \mathcal{O} \subset \mathcal{M}$, and the identity map be the chart for point $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^d$. Then it is easy to find that

$$\mathrm{Rank}(\boldsymbol{I}^{-1} \circ \boldsymbol{f} \circ \boldsymbol{\phi}) = \mathrm{Rank}(\boldsymbol{f} \circ \boldsymbol{\phi}) = \mathrm{Rank}_{\mathcal{M},\mathcal{N}}(\boldsymbol{f}). \tag{A14}$$

Then by Theorem 6, we know that

$$\dim(\boldsymbol{f}(\mathcal{X})) = \mathrm{Rank}_{\mathcal{M},\mathcal{N}}(\boldsymbol{f}) = \mathrm{Rank}(\boldsymbol{f} \circ \boldsymbol{\phi}) \leq \mathrm{Rank}(\boldsymbol{f}) = r. \tag{A15}$$

The last equal sign comes from the rank inequality of matrix multiplication $\mathrm{Rank}(\boldsymbol{AB}) \leq \min\{\mathrm{Rank}(\boldsymbol{A}), \mathrm{Rank}(\boldsymbol{B})\}$, which we will discuss later.

A.2 Proof to Theorem 1

Proof to this Theorem needs Weyl's inequalities [55] for singular values of sum of matrices.

Theorem 7 (Weyl's inequalities)**.** *Let $\boldsymbol{A}, \boldsymbol{B}$ be $p \times n$ complex matrices, $\sigma_i(\cdot)$ be the i-th largest singular value of the matrix. Then*

$$|\sigma_i(\boldsymbol{A} + \boldsymbol{B}) - \sigma_i(\boldsymbol{B})| \leq \sigma_1(\boldsymbol{B}), 1 \leq i \leq p, n. \tag{A16}$$

Let p be the number of singular values of $\boldsymbol{W}$ and $\boldsymbol{D}$. By this theorem, we have

$$\sigma_i(\boldsymbol{W}) - \delta\sigma_1(\boldsymbol{D}) \leq \sigma_i(\boldsymbol{W} + \delta\boldsymbol{D}) \leq \sigma_i(\boldsymbol{W}) + \delta\sigma_1(\boldsymbol{D}), i = 1, \cdots, p. \tag{A17}$$

To measure the numerical rank, we need to estimate the relative quantities of singular values, which are,

$$\frac{\sigma_i(\boldsymbol{W}) - \delta\sigma_1(\boldsymbol{D})}{\sigma_1(\boldsymbol{W}) + \delta\sigma_1(\boldsymbol{D})} \leq \frac{\sigma_i(\boldsymbol{W} + \delta\boldsymbol{D})}{\sigma_1(\boldsymbol{W} + \delta\boldsymbol{D})} \leq \frac{\sigma_i(\boldsymbol{W}) + \delta\sigma_1(\boldsymbol{D})}{\sigma_1(\boldsymbol{W}) - \delta\sigma_1(\boldsymbol{D})}, i = 2, \cdots, p. \tag{A18}$$

Now assume that ϵ does not belong to the following set (which is a zero measure set in $\mathbb{R}_+$)

$$\Sigma_{\boldsymbol{W}} = \left\{ \frac{\sigma_i(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})} : i = 2, \cdots, p \right\}, \tag{A19}$$

and $\mathrm{Rank}_\epsilon(\boldsymbol{W}) = r$. We know that

$$\frac{\sigma_i(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})} > \epsilon, i = 2, \cdots, r; \frac{\sigma_i(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})} < \epsilon, i = r+1, \cdots, p. \tag{A20}$$

Thus, we have that $\forall \delta < \delta_{\max}$,

$$\frac{\sigma_i(\boldsymbol{W} + \delta\boldsymbol{D})}{\sigma_1(\boldsymbol{W} + \delta\boldsymbol{D})} \geq \frac{\sigma_i(\boldsymbol{W}) - \delta\sigma_1(\boldsymbol{D})}{\sigma_1(\boldsymbol{W}) + \delta\sigma_1(\boldsymbol{D})} > \epsilon, i = 2, \cdots, r, \tag{A21}$$

$$\frac{\sigma_i(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})} \leq \frac{\sigma_i(\boldsymbol{W}) + \delta\sigma_1(\boldsymbol{D})}{\sigma_1(\boldsymbol{W}) - \delta\sigma_1(\boldsymbol{D})} < \epsilon, i = r+1, \cdots, p, \tag{A22}$$

provided that

$$\begin{aligned} \delta_{\max} = \min\{ & \frac{1}{\sigma_1(D)} \left(\frac{\sigma_r(\boldsymbol{W}) + \sigma_1(\boldsymbol{W})}{\epsilon + 1} - \sigma_1(\boldsymbol{W}) \right), \frac{\sigma_r(\boldsymbol{W})}{2\sigma_1(\boldsymbol{D})}, \\ & \frac{1}{\sigma_1(D)} \left(\sigma_1(\boldsymbol{W}) - \frac{\sigma_{r+1}(\boldsymbol{W}) + \sigma_1(\boldsymbol{W})}{\epsilon + 1} \right), \frac{\sigma_1(\boldsymbol{W})}{2\sigma_1(\boldsymbol{D})} \}. \end{aligned} \tag{A23}$$

Thus we can conclude

$$\mathrm{Rank}_\epsilon(\boldsymbol{W} + \delta\boldsymbol{D}) = \mathrm{Rank}_\epsilon(\boldsymbol{W}), \forall \delta \in [0, \delta_{\max}). \tag{A24}$$

When $\mathrm{Rank}(\boldsymbol{W}) = r < p$, it is then easy to see if $\epsilon < \frac{\sigma_r(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})}$, we have

$$\frac{\sigma_i(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})} > \epsilon, i = 2, \cdots, r; \frac{\sigma_i(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})} = 0 < \epsilon, i = r+1, \cdots, p. \tag{A25}$$

Thus setting $\epsilon_{\max} = \frac{\sigma_r(\boldsymbol{W})}{\sigma_1(\boldsymbol{W})}$, we can always have $\delta_{\max}$ acquired by Eq. (A23), such that

$$\mathrm{Rank}_\epsilon(\boldsymbol{W}) = \mathrm{Rank}(\boldsymbol{W}) = \mathrm{Rank}_\epsilon(\boldsymbol{W} + \delta\boldsymbol{D}), \forall \delta \in [0, \delta_{\max}). \tag{A26}$$

A.3 Proof to Lemma 2

Let $\boldsymbol{A}_i$ be the i-th column of matrix $\boldsymbol{A}$. Given two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times d}$, we have

$$\boldsymbol{AB} = (\boldsymbol{AB}_1, \cdots \boldsymbol{AB}_d). \tag{A27}$$

Thus for any $1 \leq i_1 < \cdots < i_K \leq d$,

$$(\boldsymbol{AB})_{i_1,\cdots,i_K} = (\boldsymbol{AB}_{i_1}, \cdots, \boldsymbol{AB}_{i_K}) = \boldsymbol{A}(\boldsymbol{B})_{i_1,\cdots,i_K}. \tag{A28}$$

By the rank theorem [24] of matrices, we have

$$\mathrm{Rank}((\boldsymbol{AB})_{i_1,\cdots,i_K}) = \mathrm{Rank}((\boldsymbol{B})_{i_1,\cdots,i_K}) - \dim(\mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}((\boldsymbol{B})_{i_1,\cdots,i_K})), \tag{A29}$$

and

$$\mathrm{Rank}(\boldsymbol{AB}) = \mathrm{Rank}(\boldsymbol{B}) - \dim(\mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}(\boldsymbol{B}). \tag{A30}$$

As $(\boldsymbol{B})_{i_1,\cdots,i_K}) \subset \boldsymbol{B}$, it is straightforward to get that

$$\mathrm{Im}((\boldsymbol{B})_{i_1,\cdots,i_K})) \subset \mathrm{Im}(B). \tag{A31}$$

Thus

$$\mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}((\boldsymbol{B})_{i_1,\cdots,i_K}) \subset \mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}(\boldsymbol{B}). \tag{A32}$$

Then we have

$$\begin{aligned}\mathrm{Rank}(\boldsymbol{B}) - \mathrm{Rank}(\boldsymbol{AB}) = \dim(\mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}(\boldsymbol{B}) &\geq \dim(\mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}((\boldsymbol{B})_{i_1,\cdots,i_K})) \\ &= \mathrm{Rank}((\boldsymbol{B})_{i_1,\cdots,i_K})) - \mathrm{Rank}((\boldsymbol{AB})_{i_1,\cdots,i_K}) \geq 0.\end{aligned} \tag{A33}$$

Note that $\mathrm{Rank}(\boldsymbol{f}_2 \circ \boldsymbol{f}_1) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}_2}\boldsymbol{J}_{\boldsymbol{f}_1})$. Then we complete the proof.

A.4 Proof to Theorem 2

The key to this principle is the rank theorem of matrices [24], which is

$$\mathrm{Rank}(\boldsymbol{AB}) = \mathrm{Rank}(\boldsymbol{B}) - \dim(\mathrm{Ker}(\boldsymbol{A}) \cap \mathrm{Im}(\boldsymbol{B})). \tag{A34}$$

Note that $\mathrm{Rank}(\boldsymbol{AB}) = \mathrm{Rank}(\boldsymbol{B}^T\boldsymbol{A}^T)$ and $\mathrm{Rank}(\boldsymbol{A}^T) = \mathrm{Rank}(\boldsymbol{A})$. Then we have

$$\begin{aligned}\mathrm{Rank}(\boldsymbol{AB}) = \mathrm{Rank}(\boldsymbol{B}^T\boldsymbol{A}^T) &= \mathrm{Rank}(\boldsymbol{A}^T) - \dim(\mathrm{Ker}(\boldsymbol{B}^T) \cap \mathrm{Im}(\boldsymbol{A}^T)) \\ &= \mathrm{Rank}(\boldsymbol{A}) - \dim(\mathrm{Ker}(\boldsymbol{B}^T) \cap \mathrm{Im}(\boldsymbol{A}^T)).\end{aligned} \tag{A35}$$

The dimension of a linear subspace will at least be zero, thus the above equations suggest

$$\mathrm{Rank}(\boldsymbol{AB}) \leq \mathrm{Rank}(\boldsymbol{B}), \mathrm{Rank}(\boldsymbol{AB}) \leq \mathrm{Rank}(\boldsymbol{A}). \tag{A36}$$

Applying this argument to the chain rule of differentials then yields the conclusion. Further using Lemma 1 gives the diminishing of intrinsic dimensions of feature manifolds.

A.5 Proof to Theorem 3

We first give the rigorous version of this theorem as follows.

Theorem 8. *Let $\boldsymbol{e}_{\boldsymbol{x}}$ be the exponential map from a small neighborhood $\mathcal{U}_{\boldsymbol{x}}$ of point $\boldsymbol{x}$ on the input feature manifold to its tangent space at $\boldsymbol{x}$, and $\boldsymbol{v}_{\boldsymbol{x}} = \boldsymbol{e}_{\boldsymbol{x}}(\boldsymbol{x})$. Let $\mathcal{X}$ be the input manifold, $s = \dim(\mathcal{X})$, $\boldsymbol{f}^i$ be the layer network, $r = \mathrm{Rank}(\boldsymbol{f}^i)$, and $\boldsymbol{f}^i(\mathcal{X})$ be the output manifold. If for almost everywhere on the input feature manifold, there is a unit vector $\boldsymbol{v} \in \boldsymbol{e}_{\boldsymbol{x}}(\mathcal{U}_{\boldsymbol{x}})$, such that the layer network $\boldsymbol{f}^i$ satisfies*

$$\lim_{t\to 0} \frac{\|\boldsymbol{f}^i \circ \boldsymbol{e}_{\boldsymbol{x}}^{-1}(\boldsymbol{v}_{\boldsymbol{x}}) - \boldsymbol{f}^i \circ \boldsymbol{e}_{\boldsymbol{x}}^{-1}(\boldsymbol{v}_{\boldsymbol{x}} + t\boldsymbol{v})\|_2}{t} = 0, \tag{A37}$$

then $\dim(\boldsymbol{f}^i(\mathcal{X})) < s$. If the number of such independent $\boldsymbol{v}$ in $\boldsymbol{e}_{\boldsymbol{x}}(\mathcal{U}_{\boldsymbol{x}})$ is k, then $\dim(\boldsymbol{f}^i(\mathcal{X})) \leq s - k$.

Now we prove this theorem.

Note that Eq. (A37) implies

$$\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}\boldsymbol{v} \in \mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}). \tag{A38}$$

As it is also easy to see

$$\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}\boldsymbol{v} \in \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}), \tag{A39}$$

we can conclude

$$\boldsymbol{0} \neq \boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}\boldsymbol{v} \in \mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}), \tag{A40}$$

where $0 \neq \boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}\boldsymbol{v}$ comes from the full rank property of exponential map and its inverse. Thus we have

$$\dim(\mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}})) \geq 1. \tag{A41}$$

Specifically, if linearly independent $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k$ satisfy Eq. (A37), we can conclude

$$\boldsymbol{0} \neq \boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}\boldsymbol{v}_i \in \mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}), i = 1, \cdots, k. \tag{A42}$$

As $\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}$ has full rank due to the property of exponential map, we know that $\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}\boldsymbol{v}_i, i = 1, \cdots, k$ are linearly independent. Then

$$\dim(\mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}})) \geq k. \tag{A43}$$

Thus the rank theorem of matrices [24] reads

$$\mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}^i \circ \boldsymbol{e}_{\boldsymbol{x}}^{-1}}) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}^i}\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}}) - \dim(\mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}})) \tag{A44}$$

$$= s - \dim(\mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{f}^i}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{e}_{\boldsymbol{x}}^{-1}})) \leq s - k. \tag{A45}$$

Combining this result with Theorem 6 proves our result.

A.6 Proof to Theorem 4

The proof to this theorem relies on the existence of Lyapunov exponents of dynamic systems. Given a linearized dynamic system

$$\dot{\boldsymbol{v}}(t) = \boldsymbol{X}_t\boldsymbol{v},\ \boldsymbol{v}(0) = \boldsymbol{v}_0 \in \mathbb{R}^n, \tag{A46}$$

its (largest) Lyapunov exponent is defined as

$$\lambda = \limsup_{t\to\infty} \frac{1}{t}\|\boldsymbol{v}\|_2. \tag{A47}$$

Further, for a sequence of subspace $\mathcal{L}_h \subset \mathcal{L}_{r-1} \subset \cdots \subset \mathcal{L}_1 \subset \mathcal{L}_0 = \mathbb{R}^n$, we can define the corresponding Lyapunov exponents of all those subspaces as

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t}\log\|\boldsymbol{v}\|_2,\ i = 1, \cdots, h+1,\ \boldsymbol{v}_0 \in \mathcal{L}_{i-1}\backslash\mathcal{L}_i, \tag{A48}$$

and we have

$$\lambda = \lambda_1 > \lambda_2 > \cdots > \lambda_h. \tag{A49}$$

It may be surprising to find that such Lyapunov exponents exist, as $\boldsymbol{v}_0$ can traverse the entire subspace $\mathcal{L}_{i-1}\backslash\mathcal{L}_i$. We will demonstrate the existence of the Lyapunov exponents for our case later in Sec. A.6.3, which is the classical results from the Furstenberg-Kesten theorem [17] and multiplicative ergodic theorem [45]. Before that, we will first assume the existence of those Lyapunov exponents for simplicity of analysis.

Now consider the case of function couplings

$$\boldsymbol{F} = \boldsymbol{f}^L \circ \cdots \circ \boldsymbol{f}^2 \circ \boldsymbol{f}^1, \tag{A50}$$

which has the Jacobian matrix

$$\boldsymbol{J}_{\boldsymbol{F}} = \boldsymbol{J}_{\boldsymbol{f}^L}\boldsymbol{J}_{\boldsymbol{f}^{L-1}} \cdots \boldsymbol{J}_{\boldsymbol{f}^2}\boldsymbol{J}_{\boldsymbol{f}^1}. \tag{A51}$$

Apparently, the following dynamic system induces the Jacobi matrix of $\boldsymbol{F}$,

$$\dot{\boldsymbol{v}}(t) = \boldsymbol{J}_{\boldsymbol{f}^t}\boldsymbol{v},\ \boldsymbol{v}(0) = \boldsymbol{v}_0 \in \mathbb{R}^n,\ t = 1, \cdots, L. \tag{A52}$$

Thus its Lyapunov exponents are given by

$$\lambda_i = \lim_{L\to\infty} \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}_0\|_2,\ i = 1, \cdots, h+1,\ \boldsymbol{v}_0 \in \mathcal{L}_{i-1}\backslash\mathcal{L}_i, \tag{A53}$$

for a chain of subspaces $\{0\} = \mathcal{L}_{h+1} \subset \mathcal{L}_h \subset \mathcal{L}_{h-1} \subset \cdots \subset \mathcal{L}_1 \subset \mathcal{L}_0 = \mathbb{R}^n$.

A.6.1 Lyapunov exponents are limits of logarithms of subspace spectral norm divided by layer depth L

We first demonstrate that the Lyapunov exponents are limits of logarithm of the spectral norm of $\boldsymbol{F}$ on $\mathcal{L}_{i-1}\backslash\mathcal{L}_i$ divided by layer depth L when $L \to \infty$, for $i = 1, ..., r$.

It is easy to see

$$\frac{1}{L} \log \|\boldsymbol{v}_0\|_2 \|\boldsymbol{J}_{\boldsymbol{F}} \frac{\boldsymbol{v}_0}{\|\boldsymbol{v}_0\|_2}\|_2 = \frac{1}{L}(\log \|\boldsymbol{v}_0\|_2 + \log \|\boldsymbol{J}_{\boldsymbol{F}} \frac{\boldsymbol{v}_0}{\|\boldsymbol{v}_0\|_2}\|_2). \tag{A54}$$

When $L \to \infty$, $\frac{1}{L} \log \|\boldsymbol{v}_0\|_2 \to 0$ for any $\boldsymbol{v}_0$, we have

$$\lambda_i = \lim_{L\to\infty} \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}_0\|_2,\ \|\boldsymbol{v}_0\|_2 = 1,\ \boldsymbol{v}_0 \in \mathcal{L}_{i-1}\backslash\mathcal{L}_i. \tag{A55}$$

Let

$$\lambda_i^L = \sup_{\|\boldsymbol{v}_0\|_2=1, \boldsymbol{v}_0\in\mathcal{L}_{i-1}\backslash\mathcal{L}_i} \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}_0\|_2. \tag{A56}$$

Note that

$$\sup_{\|\boldsymbol{v}_0\|_2=1, \boldsymbol{v}_0\in\mathcal{L}_{i-1}\backslash\mathcal{L}_i} \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}_0\|_2 = \frac{1}{L} \log \sup_{\|\boldsymbol{v}_0\|_2=1, \boldsymbol{v}_0\in\mathcal{L}_{i-1}\backslash\mathcal{L}_i} \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}_0\|_2, \tag{A57}$$

and

$$\sup_{\|\boldsymbol{v}_0\|_2=1, \boldsymbol{v}_0\in\mathcal{L}_{i-1}\backslash\mathcal{L}_i} \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}_0\|_2 = \|\boldsymbol{J}_{\boldsymbol{F}}\|_{2,i}, \tag{A58}$$

where $\|\cdot\|_{2,i}$ denote the spectral norm of a linear operator constrained on $\mathcal{L}_{i-1}\backslash\mathcal{L}_i$.Then we have

$$\lambda_i^L = \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\|_{2,i}. \tag{A59}$$

Let $\boldsymbol{e}_{i_1}, \cdots, \boldsymbol{e}_{i_k}$ be a set of standard orthogonal basis of i_k dimensional subspace $\mathcal{L}_{i-1}\backslash\mathcal{L}_i$. If the Lyapunov exponents exist, by Eq. (A55) we have for any $\epsilon > 0$, there is $N \in \mathbb{N}$ such that for all $L > N$,

$$\lambda_i - \frac{\epsilon}{2} \le \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{e}_j\|_2 \le \lambda_i + \frac{\epsilon}{2}, j = 1, \cdots, i_k. \tag{A60}$$

Let $\boldsymbol{v} = \sum_{j=1}^{i_k} \alpha_j \boldsymbol{e}_j \in \mathcal{L}_{i-1}\backslash\mathcal{L}_i$, where $\alpha_j, i = 1, \cdots, i_k, \sum_{j=1}^{i_k} \alpha_j^2 = 1$ is the coordinate of unit vector $\boldsymbol{v}$ under the basis $\boldsymbol{e}_j, j = 1, \cdots, i_k$. Assume that $\|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{e}_1\|_2 \ge \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{e}_j\|_2, j = 2, \cdots, i_k$. We

then have $\frac{|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_j\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2} \leq 1, j = 1, \cdots, i_k$, and

$$\frac{1}{L}\log\|\boldsymbol{J_F}\boldsymbol{v}\|_2 \leq \frac{1}{L}\log\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_j\|_2 \tag{A61}$$

$$= \frac{1}{L}\log\left(\frac{|\alpha_1|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2} + \sum_{j=2}^{i_k}\frac{|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_j\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2}\right)(\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2) \tag{A62}$$

$$= \frac{1}{L}\log\left(\frac{|\alpha_1|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2} + \sum_{j=2}^{i_k}\frac{|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_j\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2}\right) + \frac{1}{L}\log\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2 \tag{A63}$$

$$\leq \frac{1}{L}\log\left(1 + \sum_{j=2}^{i_k}\frac{|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_j\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2}\right) + \frac{1}{L}\log\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2 \tag{A64}$$

$$\leq \frac{1}{L}\sum_{j=2}^{i_k}\frac{|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_j\|_2}{\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2} + \frac{1}{L}\log\sum_{j=1}^{i_k}|\alpha_j|\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2 \tag{A65}$$

$$\leq \frac{1}{L}(i_k - 1) + \frac{1}{L}\log\sum_{j=1}^{i_k}|\alpha_j| + \frac{1}{L}\log\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2 \tag{A66}$$

$$\leq \frac{1}{L}(i_k - 1) + \frac{1}{2L}\log(1^2 + \cdots + 1^2)(\alpha_1^2 + \cdots + \alpha_{i_k}^2) + \frac{1}{L}\log\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2 \tag{A67}$$

$$= \frac{1}{L}(i_k - 1) + \frac{1}{2L}\log i_k + \frac{1}{L}\log\|\boldsymbol{J_F}\boldsymbol{e}_1\|_2 \tag{A68}$$

$$\leq \frac{1}{L}(i_k - 1) + \frac{1}{2L}\log i_k + \lambda_i + \frac{\epsilon}{2}. \tag{A69}$$

Thus, if we set $N_0 = \max\{N, \frac{2i_k - 2 + \log i_k}{\epsilon}\}$, then when $L > N_0$ we have $\frac{1}{L}(i_k - 1) + \frac{1}{2L}\log i_k < \frac{\epsilon}{2}$ and

$$\frac{1}{L}\log\|\boldsymbol{J_F}\boldsymbol{v}\|_2 \leq \lambda_i + \epsilon,\ \forall \boldsymbol{v} \in \mathcal{L}_{i-1}\backslash\mathcal{L}_i,\ \|\boldsymbol{v}\|_2 = 1. \tag{A70}$$

Combining Eqs. (A60) and (A70), we have for any $\epsilon > 0$, there is $N_0 \in \mathbb{N}$, such that when $L > N_0$, we always have

$$\lambda_i - \frac{\epsilon}{2} \leq \lambda_i^L = \sup_{\|\boldsymbol{v}_0\|_2 = 1, \boldsymbol{v}_0 \in \mathcal{L}_{i-1}\backslash\mathcal{L}_i}\frac{1}{L}\log\|\boldsymbol{J_F}\boldsymbol{v}_0\|_2 = \frac{1}{L}\log\|\boldsymbol{J_F}\|_{2,i} \leq \lambda_i + \epsilon. \tag{A71}$$

Thus, if the Lyapunov exponents exist, *i.e.*, the existence of limits of Eq. (A53) , we have

$$\lambda_i = \lim_{L\to\infty}\frac{1}{L}\log\|\boldsymbol{J_F}\|_{2,i} = \lim_{L\to\infty}\lambda_i^L. \tag{A72}$$

A.6.2 Singular value distributions of Jacobian matrices of deep function coupling

In Sec. A.6.1 we have proved that the Lyapunov exponents (if they exist) are limits of logarithms of subspace spectral norms divided by L. Here we use this property to prove the deficiency of numerical ranks, *i.e.*, Eq. (9).

We first introduce the Courant-Fischer min-max theorem [24] of sigular values.

Theorem 9 (Courant-Fischer Min-max Theorem)**.** *Let $\boldsymbol{A}$ be a $d \times n$ complex matrix and $\sigma_i(\boldsymbol{A})$ denote its i-th largest singular value, $i = 1, \cdots, \min\{d, n\}$. Then we have*

$$\sigma_i(\boldsymbol{A}) = \sup_{\dim(\mathcal{V})=i}\inf_{\boldsymbol{v}\in\mathcal{V}, \|\boldsymbol{v}\|_2=1}\|\boldsymbol{A}\boldsymbol{v}\|_2, \tag{A73}$$

$$\sigma_i(\boldsymbol{A}) = \inf_{\dim(\mathcal{V})=n-i+1}\sup_{\boldsymbol{v}\in\mathcal{V}, \|\boldsymbol{v}\|_2=1}\|\boldsymbol{A}\boldsymbol{v}\|_2, \tag{A74}$$

$$\tag{A75}$$

where $\mathcal{V}$ traverses subspaces of $\mathbb{R}^n$.

This theorem also serves as one of the definitions to singular values.

Now assume that $\dim(\mathcal{L}_0 \backslash \mathcal{L}_1) = r$, and we consider only the case of $d = n$ for simplicity. For any $\epsilon > 0$, we have $N \in \mathbb{N}$ such that when $L > N$,

$$\inf_{\boldsymbol{v} \in \mathcal{L}_0 \backslash \mathcal{L}_1, \|\boldsymbol{v}\|_2 = 1} \|\boldsymbol{J_F} \boldsymbol{v}\|_2 \geq \exp L(\lambda_1 - \epsilon) \tag{A76}$$

$$\tag{A77}$$

due to Eq. (A55). As $\dim(\mathcal{L}_0 \backslash \mathcal{L}_1) = r$, by Theorem 9, we have

$$\sigma_1(\boldsymbol{J_F}) \geq \cdots \geq \sigma_r(\boldsymbol{J_F}) \geq \exp L(\lambda_1 - \epsilon). \tag{A78}$$

For $\boldsymbol{v} \in \mathcal{L}_0 = \mathbb{R}^n$ and $\|\boldsymbol{v}\|_2 = 1$, as $\mathcal{L}_0 = \mathcal{L}_0 \backslash \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_h \backslash \mathcal{L}_{h+1}$ ($\oplus$ denotes direct sum of linear quotient subspaces in the Banach space), there is $\boldsymbol{v}_i \in \mathcal{L}_{i-1} \backslash \mathcal{L}_i, i = 1, \cdots, h+1$, such that

$$\|\boldsymbol{v}_1\|_2^2 + \cdots + \|\boldsymbol{v}_{h+1}\|_2^2 = 1 \tag{A79}$$

and

$$\boldsymbol{v} = \boldsymbol{v}_1 + \cdots + \boldsymbol{v}_{h+1}. \tag{A80}$$

Then by the conclusion of Sec. A.6.1, we have

$$\limsup_{L \to \infty} \|\boldsymbol{J_F} \boldsymbol{v}\|_2 \leq \limsup_{L \to \infty} \|\boldsymbol{J_F} \boldsymbol{v}_1\|_2 + \cdots + \limsup_{L \to \infty} \|\boldsymbol{J_F} \boldsymbol{v}_{h+1}\|_2 \tag{A81}$$

$$\leq \|\boldsymbol{v}_1\|_2 \lim_{L \to \infty} \|\boldsymbol{J_F}\|_{2,2} + \cdots + \|\boldsymbol{v}_{h+1}\|_2 \lim_{L \to \infty} \|\boldsymbol{J_F}\|_{2,h+1} \leq \sum_{i=1}^{h+1} \|\boldsymbol{v}_i\|_2 \lambda_i \leq \lambda_1. \tag{A82}$$

Thus there is $N_1 \in \mathbb{N}$, such that when $L > N_1$, we have

$$\sup_{\boldsymbol{v} \in \mathcal{L}_0, \|\boldsymbol{v}\|_2 = 1} \|\boldsymbol{J_F} \boldsymbol{v}\|_2 \leq \lambda_1 + \epsilon. \tag{A83}$$

As $\dim(\mathcal{L}_1) = n - 1 + 1$, by Theorem 9, we have when $L > N_1$,

$$\sigma_r(\boldsymbol{J_F}) \leq \cdots \leq \sigma_1(\boldsymbol{J_F}) \leq \exp L(\lambda_1 + \epsilon). \tag{A84}$$

In conclusion, when $L > N_0 = \max\{N, N_1\}$, we have

$$\exp L(\lambda_1 - \epsilon) \leq \sigma_1(\boldsymbol{J_F}) \leq \exp L(\lambda_1 + \epsilon). \tag{A85}$$

Thus when $L \to \infty$, we have

$$\sigma_1(\boldsymbol{J_F}) \sim \exp L\lambda_1. \tag{A86}$$

Using the same argument for $\sigma_2(\boldsymbol{J_F}), \cdots, \sigma_n(\boldsymbol{J_F})$, we can find that if let $\hat{\lambda}_1 \geq \hat{\lambda}_2 \cdots \geq \hat{\lambda}_n$ be the Lyapunov exponents counting repetitions, *i.e.*,

$$\hat{\lambda}_k = \lambda_i, \text{ if } \sum_{j=1}^{i-1} \dim(\mathcal{L}_{j-1} \backslash \mathcal{L}_j) < k \leq \sum_{j=1}^{i} \dim(\mathcal{L}_{j-1} \backslash \mathcal{L}_j), i = 1, \cdots, h+1, \tag{A87}$$

then

$$\sigma_i(\boldsymbol{J_F}) \sim \exp L\hat{\lambda}_i. \tag{A88}$$

Note that

$$\hat{\lambda}_1 = \cdots = \hat{\lambda}_r = \lambda_1, \hat{\lambda}_i \leq \lambda_2 < \lambda_1, i = r+1, \cdots, n. \tag{A89}$$

Thus we have

$$\frac{\sigma_i(\boldsymbol{J_F})}{\sigma_1(\boldsymbol{J_F})} \sim \exp L(\hat{\lambda}_i - \hat{\lambda}_1) \to 0, i = r+1, \cdots, n. \tag{A90}$$

As a consequence, $\mathrm{Rank}_\epsilon(\boldsymbol{F}) \leq r$ for any $\epsilon > 0$ when $L \to \infty$.

A.6.3 Existence of Lyapunov exponents for Jacobian matrices of deep function coupling

In above analysis, we have proven Theorem 4 under the existence of Lyapunov exponents. In this section, we introduce the classical result of multiplicatve ergodic theorem in the specific domain of random matrices, which is proposed by Furstenberg and Kesten [17, 18].

Theorem 10 (Multiplicatve Ergodic Theorem (Theorem 3.9 of [18])). *Let μ be a probability measure on all convertible matrices of $\mathbb{R}^{n\times n}$ which satisfies*

$$\mathbb{E}_{\mu}[\max\{\log \|\boldsymbol{J}_{\boldsymbol{f}^k}^{\pm 1}\|_2, 0\}] < \infty,\ k = 1, \cdots, L. \tag{A91}$$

If each $\boldsymbol{J}_{\boldsymbol{f}^k}$ independently follows μ, then we have a chain of subspaces $\{0\} = \mathcal{L}_{h+1} \subset \mathcal{L}_h \subset \cdots \subset \mathcal{L}_1 \subset \mathcal{L}_0 = \mathbb{R}^n$ and corresponding postive real constants $\lambda_1 > \lambda_2 > \cdots > \lambda_{h+1}$ such that almost surely

$$\lambda_i = \lim \frac{1}{t} \log \|\boldsymbol{J}_{\boldsymbol{F}}\boldsymbol{v}\|_2,\ \forall \boldsymbol{v} \in \mathcal{L}_{i-1} \backslash \mathcal{L}_i,\ i = 1, \cdots, h+1, \tag{A92}$$

which means the existence of the Lyapunov exponents.

Combining this theorem and the arguments above, we can finally prove Theorem 4.

A.7 Proof to Theorem 5

This theorem can be deduced from the Lyapunov components of Ginibre matrices (polynomial ensemble of square matrices sampled *i.i.d* from standard Gaussian).

Theorem 11 (Exact Lyapunov Exponent Distribution for Ginibre Matrices [41]). *If μ in Theorem 10 is standard Gaussian, then $h + 1 = n$, and*

$$\lambda_i = \log\left(2 + \psi(\frac{n-i+1}{2})\right), i = 1, \cdots, n. \tag{A93}$$

Combining this theorem with Theorem 4 can directly yield our result.

A.8 Proof to Corollary 1

This theorem is the direct result of Theorems 4 and 11. Note that it is easy to get

$$\lambda_1 = \lim_{L\to\infty} \frac{1}{L} \log \|\boldsymbol{J}_{\boldsymbol{F}}\|_2 \tag{A94}$$

for standard Gaussian μ.

B Influences of Structures

Skip Connection Skip Connection is the most direct method to solve rank diminishing at the initialization period. In our formulation, the definition of a layer network requires it to accept inputs purely from its predecessor layer as

$$\boldsymbol{x}^i = \boldsymbol{f}^i(\boldsymbol{x}^{i-1}),\ \boldsymbol{x}^{i-1} = \boldsymbol{f}^{i-1}(\boldsymbol{x}^{i-2}). \tag{A95}$$

However, when we add a skip connection from its ancestor layer $\boldsymbol{f}^s, s < i-1$, we have

$$\boldsymbol{x}^i = \boldsymbol{f}^i(\boldsymbol{x}^{i-1}, \boldsymbol{x}^s), \boldsymbol{x}^{i-1} = \boldsymbol{f}^{i-1}(\boldsymbol{x}^{i-2}), \boldsymbol{x}^{i-2} = \boldsymbol{f}^{i-2} \circ \cdots \circ \boldsymbol{f}^s(\boldsymbol{x}^{s-1}), \boldsymbol{x}^s = \boldsymbol{f}^s(\boldsymbol{x}^{s-1}). \tag{A96}$$

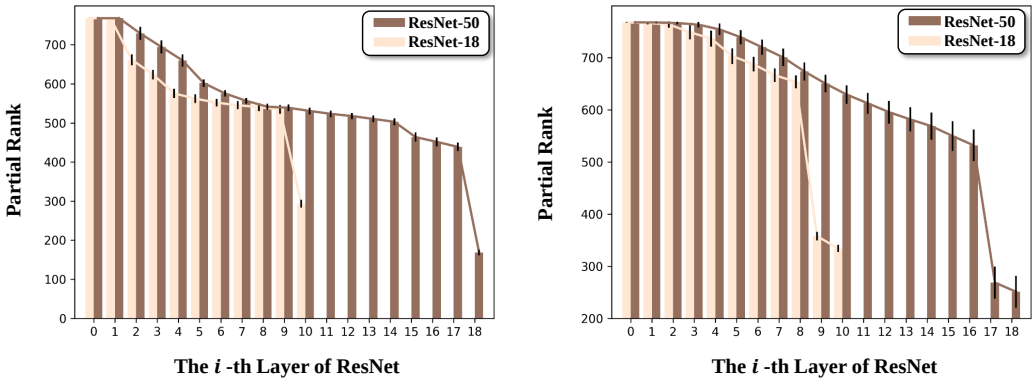
It actually makes the coupling of layers

$$\hat{\boldsymbol{f}}^s = \begin{pmatrix} \boldsymbol{f}^{i-1} \circ \cdots \circ \boldsymbol{f}^s \\ \boldsymbol{f}^s \end{pmatrix}, \tag{A97}$$

the true predecessor layer to $\boldsymbol{f}^i$, as

$$\boldsymbol{x}^i = \boldsymbol{f}^i(\hat{\boldsymbol{x}}),\ \hat{\boldsymbol{x}} = \hat{f}^s(\boldsymbol{x}^{s-1}). \tag{A98}$$

Thus the true layer depth is cut down by $i - s$, remaining $L - (i - s)$ layers. Skip connection is usually used with the residual network. This structure can ease rank diminishing inside the layer $\hat{f}^s$, which we will discuss later. Overall, skip connection shortens the length of the chain of Jacobian matrices, thus restraining rank diminishing.



(a) Rank Histogram for modified ResNet **(b) Rank Histogram for vanilla ResNet**

Figure A5: The partial rank of Jacobian matrices and perturbed PCA dimensions at the i-th layer of the modified ResNet-18 and ResNet-50 on ImageNet. We remove the operation of downsampling so that the feature dimension of the modified ResNet does not change (*e.g.*, $H \times W \times C = 802,816$ for ResNet-50). (a) The results of the modified ResNet-18 and ResNet-50. (b) The results of the initial ResNet-18 and ResNet-50. All the results are measured before training and using random initialization.

BatchNorm Some previous works [13, 5] discuss the role of BatchNorm in restraining rank diminishing. They show that BatchNorm may slow down the speed of rank diminishing in neural networks in some specific cases.

Residual Network Residual Network is another useful tool to restrain rank diminishing at the initialization period. The residual network $\boldsymbol{r}$ has the form

$$\boldsymbol{x}^o = \boldsymbol{r}(\boldsymbol{x}^i) = \boldsymbol{x}^i + \mathrm{Res}(\boldsymbol{x}^i), \tag{A99}$$

where $\boldsymbol{x}^o$ and $\boldsymbol{x}^i$ are the output feature and input feature, respectively. Usually the residual term $\mathrm{Res}(\boldsymbol{x}^i)$ is small compared with the input $\boldsymbol{x}^i$ at the initialization period, as is pointed out by related works [20]. Assume that

$$\|\boldsymbol{J}_{\mathrm{Res}}\|_2 < \epsilon, \tag{A100}$$

where ϵ is very small. Then we have

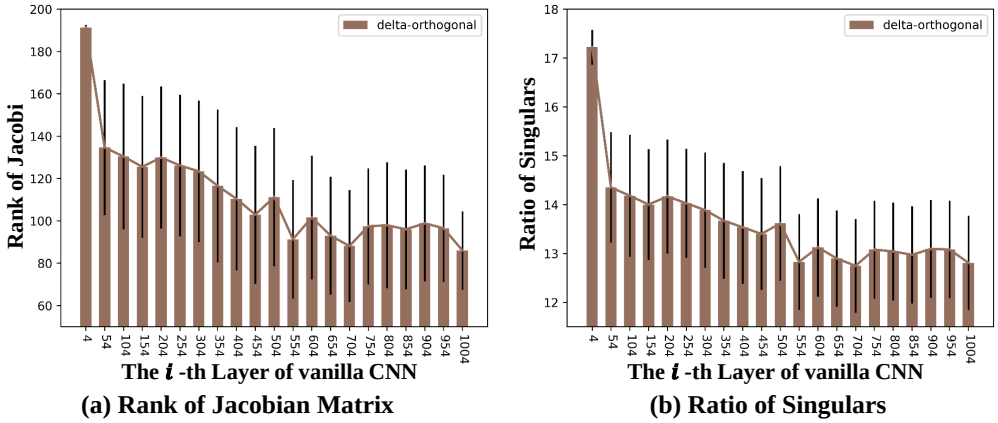
$$\boldsymbol{J}_{\boldsymbol{r}} = \boldsymbol{I} + \boldsymbol{J}_{\mathrm{Res}} \tag{A101}$$

is a diagonally dominant matrix, thus it has full rank. This means its kernel space $\mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{r}}) = \{0\}$ is a zero dimension space. Thus by the Rank Theorem, for any predecessor layer $\boldsymbol{f}$, $\boldsymbol{r} \circ \boldsymbol{f}$ will not lose rank as

$$\begin{aligned} \mathrm{Rank}(\boldsymbol{r} \circ \boldsymbol{f}) = \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{r}}\boldsymbol{J}_{\boldsymbol{f}}) &= \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}}) - \dim(\mathrm{Ker}(\boldsymbol{J}_{\boldsymbol{r}}) \cap \mathrm{Im}(\boldsymbol{J}_{\boldsymbol{f}})) \\ &= \mathrm{Rank}(\boldsymbol{J}_{\boldsymbol{f}}) = \mathrm{Rank}(\boldsymbol{f}). \end{aligned} \tag{A102}$$

However, in the well-trained ResNet18 and ResNet50 networks, we still observe considerable diminishing of ranks in Fig. 1. The reason for this phenomenon could be that, during training the magnitude of the residual term $\mathrm{Res}(\boldsymbol{x})$ becomes large. Then the argument above no longer stand and hence the rank becomes lower. Taking the 16-th layer of the ResNet50 as example, we find that the rank of this layer drops from 530 to 119 after training, while the relative magnitude $\frac{\|\mathrm{Res}(\boldsymbol{x})\|_2}{\|\boldsymbol{x}\|_2}$ increases from 0.5127 to 0.9557 after training. This may explain why the residual connection is less effective in preventing network ranks after training.

Influences of the Pooling Layers and Width To better validate the rank behavior of deep neural networks (*e.g.*, ResNet), we remove the operation of downsampling in the ResNet so that the feature dimension (*e.g.*, $x \in \mathbb{R}^{H \times W \times C = 802,816}$ for ResNet-50) will not change. This modified ResNet can exclude the effect of pooling layers and changes of layer width. As shown in Fig.A5, we



(a) Rank of Jacobian Matrix **(b) Ratio of Singulars**

Figure A6: The rank of Jacobi matrices and perturbed PCA dimensions at the i-th layer of the 10,000 CNN initialized by Delta-Orthogonal on CIFAR-10.

show the partial rank of Jacobian matrices and perturbed PCA dimensions at the i-th layer of the modified ResNet-18 and ResNet-50 on ImageNet. We can find that the curves of partial ranks and perturbed PCA dimensions share a similar and consistent trend of decreasing as the initial networks. The consistent behavior of partial ranks and perturbed PCA dimensions also shows a monotonic decreasing property of network ranks. Thus the overall trend of rank diminishing seems to be independent of the pooling layers and changes of width. On the other hand, the partial rank witnesses a considerable drop near the terminal layer after applying pooling layers, which means it does have an effect on the network ranks.

C Orthogonal Initialization

Previous work [60] has demonstrated that under carefully designed initialization rule, we can train very deep (1,0000 layer) plain CNNs. It is then curious to investigate that whether the phenomenon of rank diminishing happens in this case. To this end, we empirically measure the numerical rank of Jacobi matrices and perturbed PCA dimensions of an extremely deep CNN initialized by the Delta-Orthogonal method [60]. While computing the Jacobi matrices of very deep networks is infeasible in time, we only compute the 4-1,004 layer of the network. We omit the first 4 layers as there are downsampling architectures. We measure two metrics of each layer:

1. the numerical rank of the Jacobi matrix;
2. the ratio between sum of small singular values and large singular values $\frac{\sum_{i=11}^{3072}\sigma_i}{\sum_{i=1}^{1}0\sigma_i}$, where 3,072 is the width of this network, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{3072}$ are singular values of the Jacobi matrix.

As shown in FigA6, a generally decreasing trend of the Jacobi rank and singular ratio can be observed. This result is consistent with Theorem 4. Although orthogonal initialization can suppress rank decrease to a certain extent, the impact of low rank cannot be ignored when the network is deep enough.

D Code

Algorithm A1 provides the pseudo-code of partial rank of the Jacobian. The implementation of the Algorithm A1 can refer to the 'rank_jacobian.py' python file.

Algorithm A2 provides the pseudo-code of perturbed PCA dimension of feature spaces. The implementation of the Algorithm A2 can refer to the 'rank_perturb.py' python file.

Arch.	Network	Activ.	#Param.	Main Block	#Layer	Top-1 Acc.
ResNets	ResNet-18 [21]	ReLU [39]	11.7M	Bottleneck	11	69.8%
	ResNet-50 [21]	ReLU [39]	25.6M	Bottleneck	19	76.1%
MLP-like	GluMixer-24 [47]	SiLU [22]	25.0M	Mixer-Block	24	78.1%
	ResMLP-S24 [51]	GELU [22]	30.0M	Mixer-Block	24	79.4%
Transformer	ViT-T [16]	GELU [22]	5.7M	ViT-Block	13	75.5%
	Swin-T [35]	GELU [22]	29.0M	Swin-Block	18	81.3%

Table A2: Information of networks used in empirical validations. All pretrained on ImageNet.

Algorithm A3 provides the pseudo-code of the classification dimension. The implementation of the Algorithm A3 can refer to the 'run_cls_dim.py' python file.

Algorithm A4 provides the pseudo-code of independence deficit. The implementation of the Algorithm A4 can refer to the 'run_deficit.py' python file.

E Experiments Setup

Information of those networks used in validations is listed in Tab. A2. When measuring rank, we set $\epsilon = \text{eps} \times N$, where eps is the digital accuracy of float32 (*i.e.*, $1.19e-7$) and N is the number of singular values of the matrix to measure. This threshold represents the minimum digital accuracy of numerical rank we can capture in data stored as float32. All the experiments are conducted on the validation set of ImageNet and NVIDIA A100-SXM-80G GPUs.

F Partial Rank of the Jacobian: Estimating Lower Bound of Lost Rank in Deep Networks

To enable the validation of trend of the network ranks, we propose to compute only the rank of sub-matrices of the Jacobian as an alternative. Those sub-matrices are also the Jacobian matrices with respect to a fixed small patch of inputs. Rigorously, given a function $\boldsymbol{f}$ and its Jacobian $\boldsymbol{J_f}$, we denote partial rank of the Jacobian as the rank of a sub-matrix of the Jacobian that consists of the j_1-th, j_2-th,...,j_K-th column of the original Jacobian

$$\text{PartialRank}(\boldsymbol{J_f}) = \text{Rank}(\text{Sub}(\boldsymbol{J_f}, j_1, ..., j_K)) = \text{Rank}((\partial \boldsymbol{f}_i / \partial \boldsymbol{x}_{j_k})_{d \times K}), \tag{A103}$$

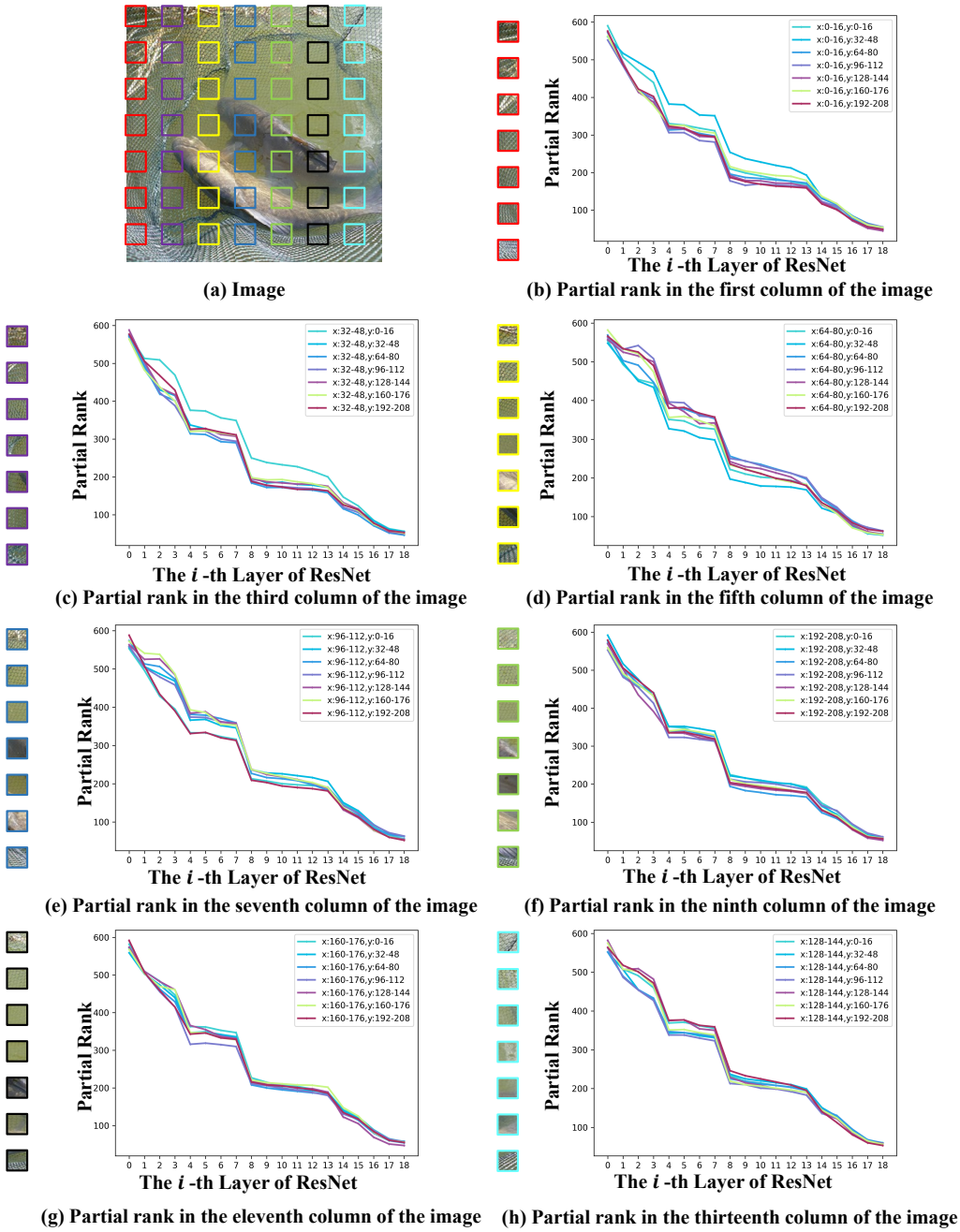
where $1 \leq j_1 < \ldots < j_K \leq n$. We can efficiently compute sub-matrix of the Jacobian by zero padding to small patches of input images. For any data point $\boldsymbol{x} \in \mathbb{R}^n$, let $\text{Sub}(\boldsymbol{x}, j_1, ..., j_K) = (\boldsymbol{x}_{j_1}, ..., \boldsymbol{x}_{j_K})^T \in \mathbb{R}^K$, and $\boldsymbol{\psi}$ pad $\text{Sub}(\boldsymbol{x}, j_1, ..., j_K)$ to the spatial size of $\boldsymbol{x}$ with zeros: $\boldsymbol{\psi}(\text{Sub}(\boldsymbol{x}, j_1, ..., j_K)) = (0, .., 0, \boldsymbol{x}_{j_1}, 0, ..., \boldsymbol{x}_{j_K}, 0, ..., 0)^T \in \mathbb{R}^n$ with $\boldsymbol{\psi}(\text{Sub}(\boldsymbol{x}, j_1, ..., j_K))_{j_k} = \boldsymbol{x}_{j_k}, k = 1, ..., K$. We then have $\boldsymbol{J}_{\boldsymbol{f} \circ \boldsymbol{\psi}} = \text{Sub}(\boldsymbol{J_f}, j_1, ..., j_K)$. As K can be very small compared with n, computing $\boldsymbol{J}_{\boldsymbol{f} \circ \boldsymbol{\psi}}$ can be very cheap in time and space. The partial rank of Jacobian matrices of the network layers measures information captured among the spatial footprint $j_1, ..., j_K$ of the original input. They inherit the order relation of the rank of full Jacobian matrices. Thus we can validate the rank diminishing of network Jacobian matrices through the partial rank.

Lemma 2. *For differentiable* $\boldsymbol{f}_1, \boldsymbol{f}_2$, $|\text{Rank}(\boldsymbol{f}_1) - \text{Rank}(\boldsymbol{f}_2 \circ \boldsymbol{f}_1)| \geq |\text{Rank}(\text{Sub}(\boldsymbol{f}_1, j_1, \ldots, j_K)) - \text{Rank}(\text{Sub}(\boldsymbol{f}_2 \circ \boldsymbol{f}_1, j_1, \ldots, j_K))|, \forall 1 \leq K \leq n, 1 \leq j_1, \ldots, j_K \leq n$. *Thus variance of partial ranks of adjacent sub-networks gives a lower bound on the variance of their ranks.*

F.1 Partial Rank of Jacobians under Different Input Patches

In Fig. A7 we report partial ranks of different input image patches (marked with colored boxes in Fig. A7(a)) for the layers of ResNet-50 on ImageNet. We can find that the curves of partial ranks share a similar and consistent trend among different input patches. Thus, picking one patch, for example, the central patch of $16 \times 16 \times 3$ pixels we use in Sec. 6.1, could be enough to demonstrate

the overall behavior of network ranks. The consistent behavior of all those partial ranks also shows that partial rank is a good tool to investigate network ranks.



(a) Image (b) Partial rank in the first column of the image (c) Partial rank in the third column of the image (d) Partial rank in the fifth column of the image (e) Partial rank in the seventh column of the image (f) Partial rank in the ninth column of the image (g) Partial rank in the eleventh column of the image (h) Partial rank in the thirteenth column of the image

Figure A7: The partial ranks of different input patches at the i-th layer of ResNet-50 on ImageNet.

G Estimating Dimension Diminishing in Features

Measuring the intrinsic dimension of feature manifolds is known to be hard. However, we manage to give a rough estimation to the dimension dropped by different layer networks. To do this, we use a new metric called the Perturbed PCA Dimension. It measures the expectation of PCA dimension of small local neighborhoods over the feature manifold.

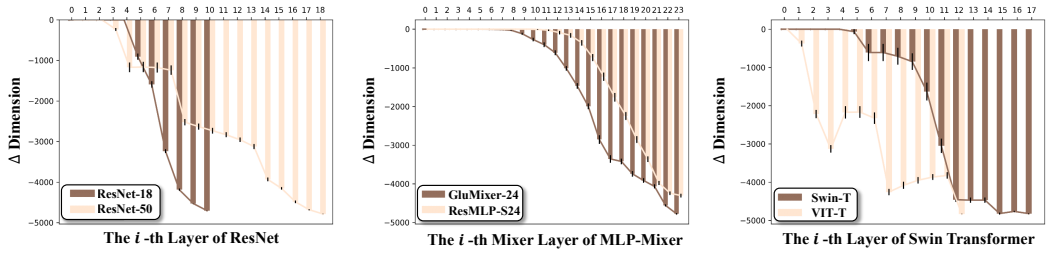
Let $\boldsymbol{F}_k$ be the k-th sub-network of the whole network $\boldsymbol{F}$. We want to measure the Perturbed PCA Dimension of $\boldsymbol{F}_k(\mathcal{X})$, where $\mathcal{X}$ is the input data domain. To this end, we compute

$$\text{PertDim} = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{\mathcal{X}}}[\text{PCADim}(\{\boldsymbol{F}_k(\boldsymbol{x}+\boldsymbol{\epsilon}) : \boldsymbol{\epsilon} \sim \mathcal{N}(\mathcal{O}, \delta\mathcal{I})\})], \tag{A104}$$

where PCADim for a set is the number of PCA eigenvalues larger than a threshold ξ. For each point $\boldsymbol{x}$, we sample 50,000 different perturbation $\boldsymbol{\epsilon}$ to compute the PCA dimension of the neighborhood of $\boldsymbol{F}_k(\boldsymbol{x})$. When computing the PCA dimension, we set $\delta = 1e-3$ and $\xi = 1.19e-7 \times 50000 \times \text{eig}_{\max}$, where $\text{eig}_{\max}$ is the largest PCA eigenvalue. We then compute the mean value of PCA dimensions over the neighborhood of 100 random samples in the validation set of ImageNet as the final result.

We do not use PCA dimension of the feature manifolds directly as it is unable to cope with the highly non-linear structure of intermediate feature manifolds. However, the Perturbed PCA Dimension is able to estimate the dimensions of local neighborhoods of points in the feature manifolds. As local neighborhoods can be viewed as linear if the network is smooth, the Perturbed PCA Dimension could be more feasible than PCA dimension in our case. We provide the pseudo-code to compute the Perturbed PCA Dimension in Algorithm A2.

However, the perturbation is made in the ambient space of the input data manifold $\mathcal{X}$ rather than the data manifold itself. Thus this estimation may considerably overestimate the intrinsic dimensions of feature manifolds. So we merely care about how many Perturbed PCA Dimensions are lost by a sub-network instead of its own Perturbed PCA Dimension. We call this quantity Δ Dimension, which is the difference between the Perturbed PCA Dimension of the current layer and that of the input layer for the given deep network. As shown in Fig. A8, we show the dropped dimensions of different feature layers of the CNN, MLP, and Transformer architectures on ImageNet. The results show that the Perturbed PCA Dimensions of feature manifolds of most networks decrease as the networks get deeper, thus confirming the rank diminishing principle we propose in Theorem 2.



(a) Dimension Histogram for ResNet (b) Dimension Histogram for MLP-Mixer (c) Dimension Histogram for Transformer

Figure A8: Dropped Perturbed PCA Dimension of different layers. Δ Dimension for the i-th layer is the difference between the Perturbed PCA Dimension of the i-th layer and that of the input layer of the CNN, MLP, and Transformer architectures on ImageNet.

H More Examples of Independence Deficit

We provide more examples of independence deficit, which shows that classification confidence of some categories can be lineally decided by a few other categories with fixed coefficients. The results obtained by ResNet-18, ResNet-50, GluMixer-24, ResMLP-S24, ViT-T, and Swin-T are reported in Fig. A9 - Fig. A14, respectively. All the results are obtained by solving the Lasso problem in Sec. 6.2. In each figure, we report the classification accuracy for category 'i': the accuracy by calculating logits with Eq. (12) is reported as 'acc.'; and the original model accuracy is reported as 'ori. acc.'. Both the metrics are measured in the whole ImageNet validation set. We further report the classification accuracy on positive samples only for both metrics as 'pos' following 'acc.' and 'ori. acc.' correspondingly. The results show a universal independence deficit phenomenon for broad categories in all those deep networks.

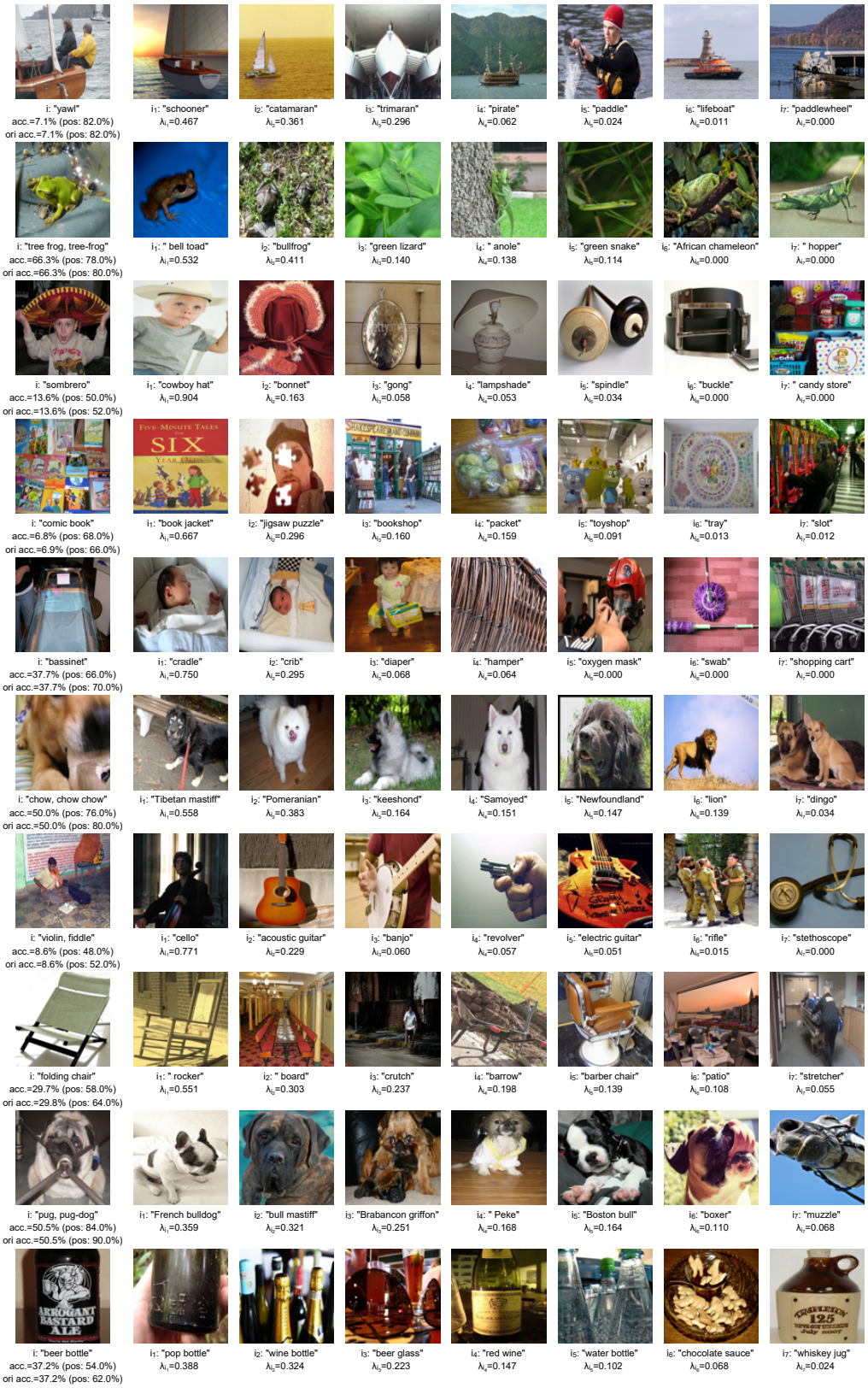


Figure A9: Results from ResNet-18, where 'acc.' and 'ori acc.' denote the classification accuracies on the ImageNet validation set, while 'pos: xx%' is the accuracy on positive samples only.

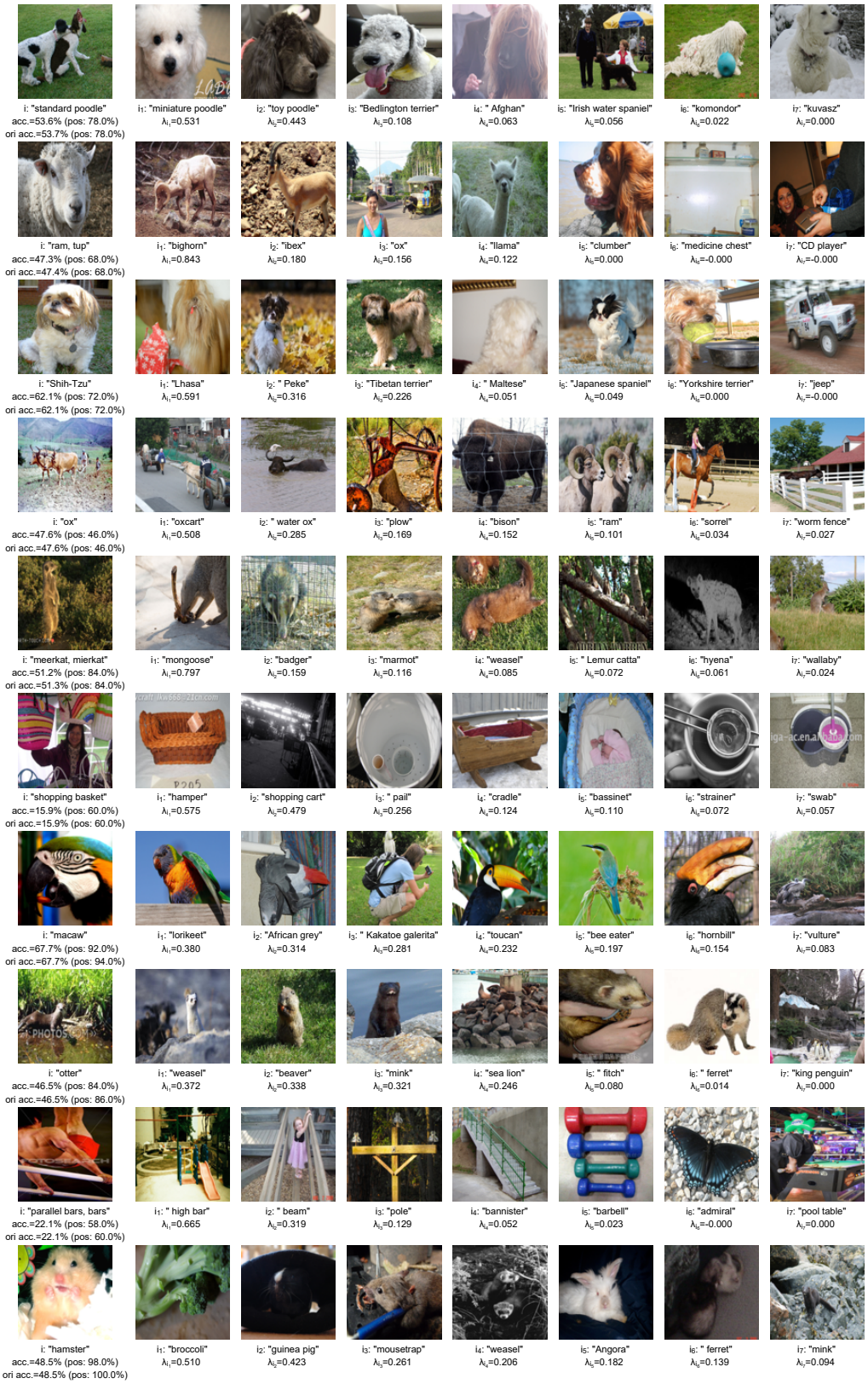


Figure A10: Results from ResNet-50, where 'acc.' and 'ori acc.' denote the classification accuracies on the ImageNet validation set, while 'pos: xx%' is the accuracy on positive samples only.

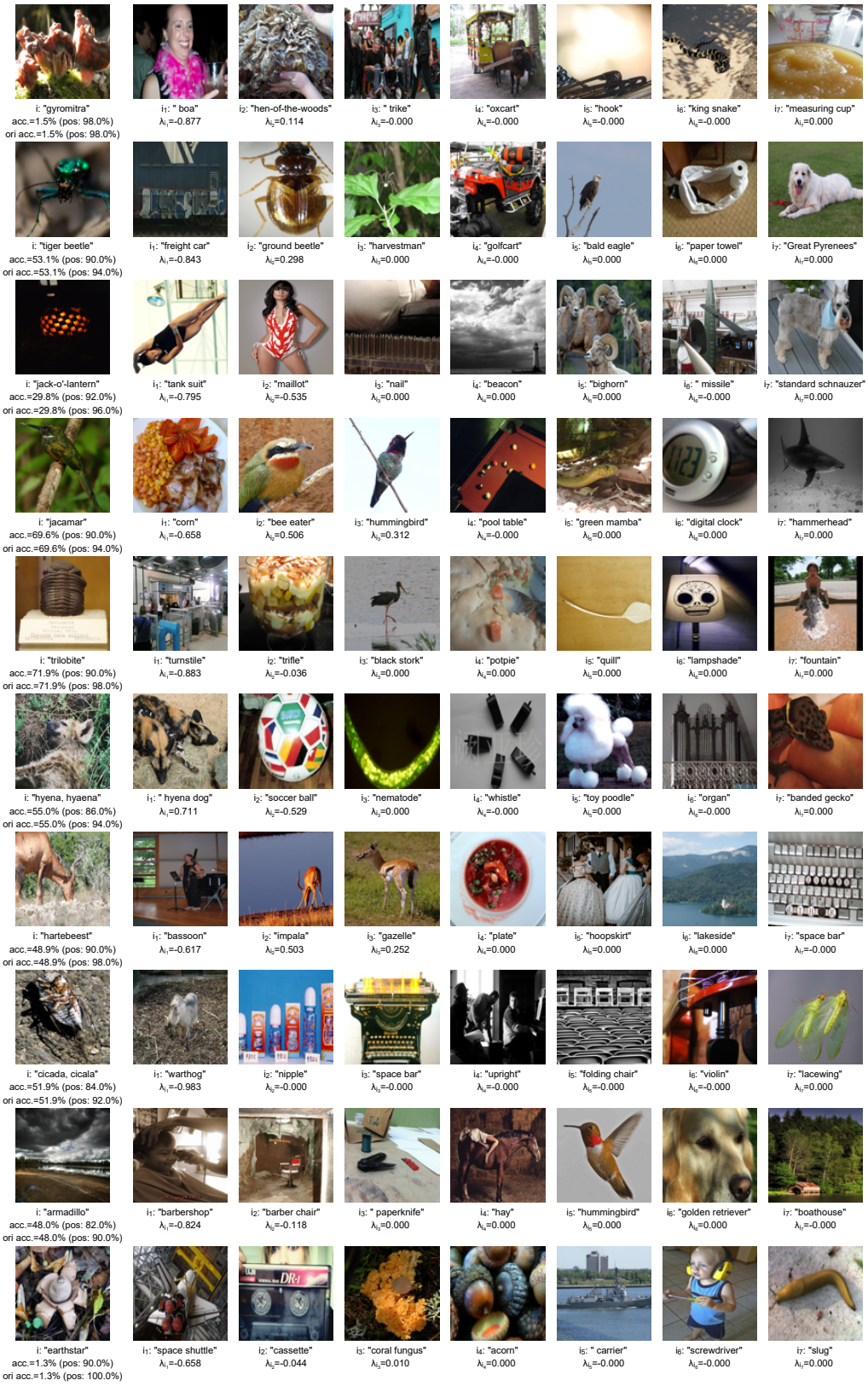


Figure A11: Results from GluMixer-24, where 'acc.' and 'ori acc.' denote the classification accuracies on the ImageNet validation set, while 'pos: xx%' is the accuracy on positive samples only.

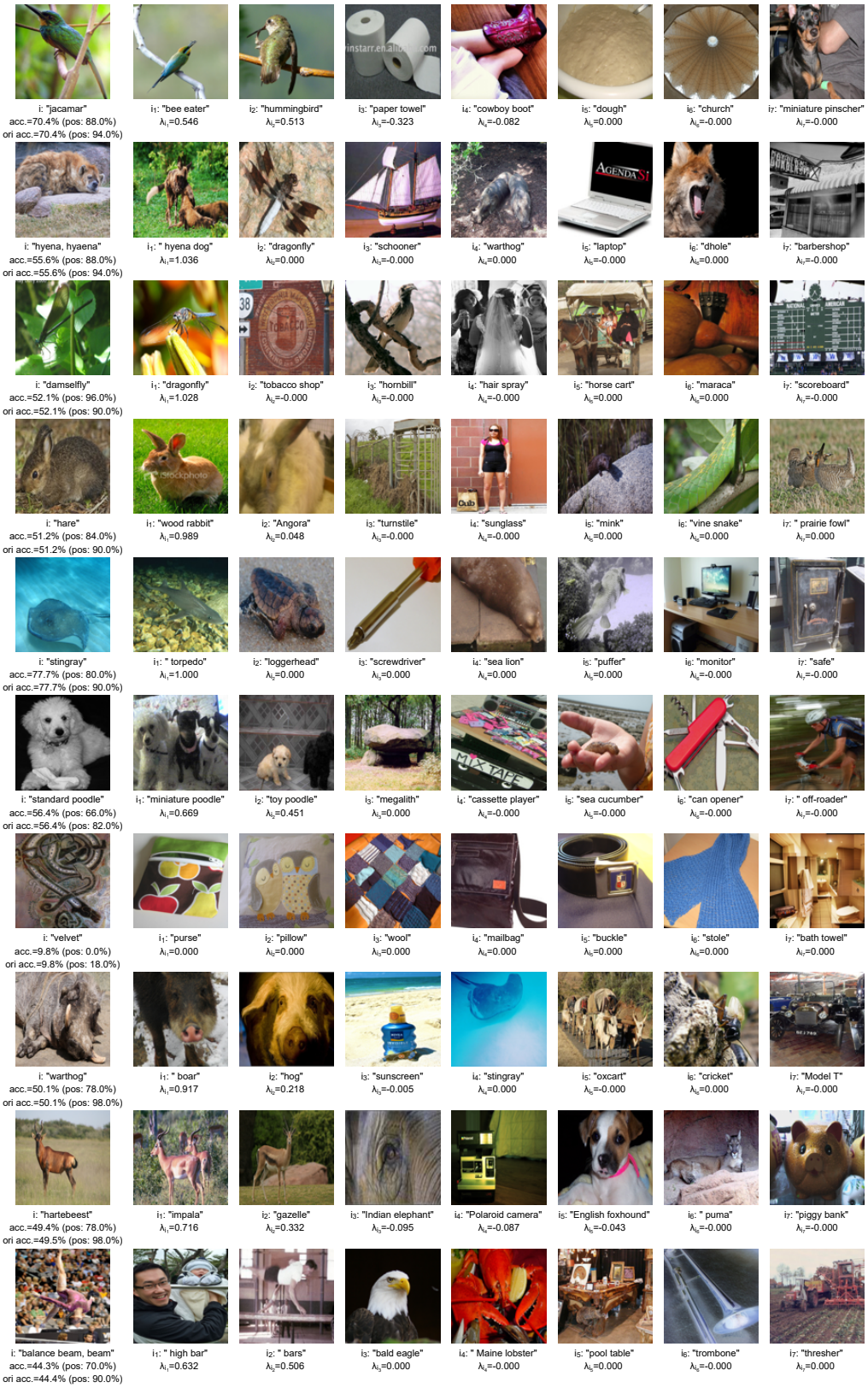


Figure A12: Results from ResMLP-S24, where 'acc.' and 'ori acc.' denote the classification accuracies on the ImageNet validation set, while 'pos: xx%' is the accuracy on positive samples only.

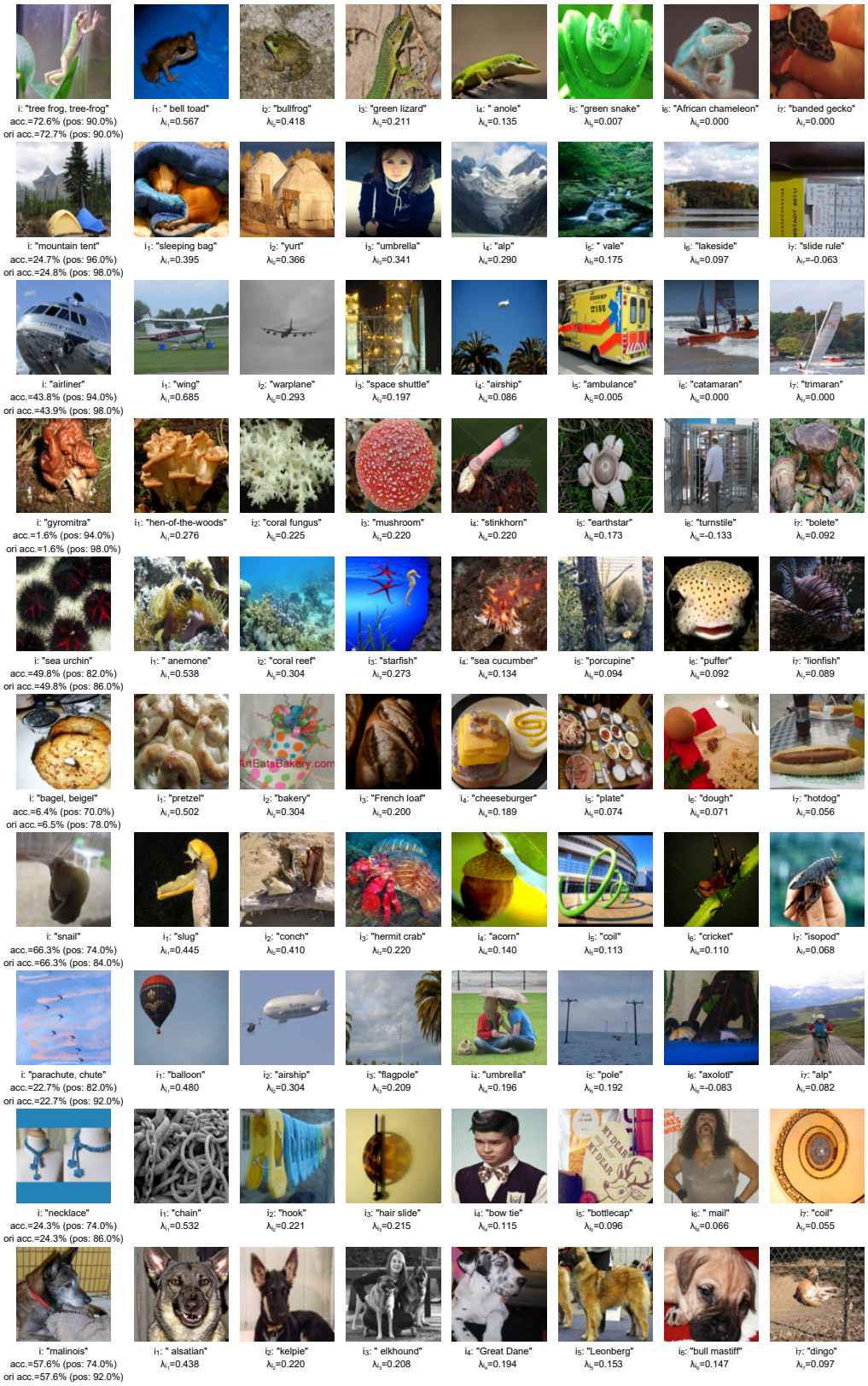


Figure A13: Results from ViT-T, where 'acc.' and 'ori acc.' denote the classification accuracies on the ImageNet validation set, while 'pos: xx%' is the accuracy on positive samples only.

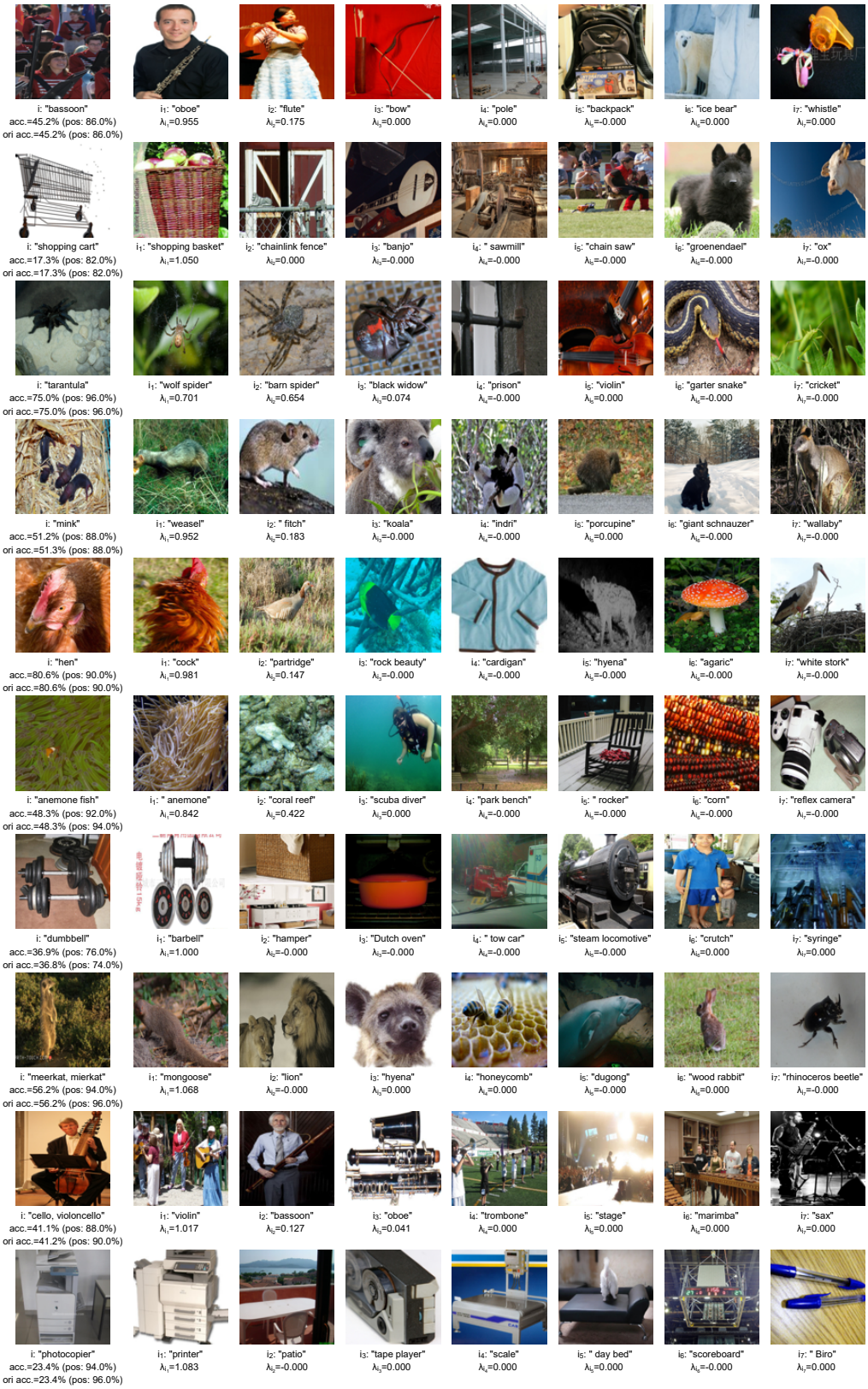


Figure A14: Results from Swin-T, where 'acc.' and 'ori acc.' denote the classification accuracies on the ImageNet validation set, while 'pos: xx%' is the accuracy on positive samples only.

Algorithm A1 Pseudocode of Partial Rank of the Jacobian.

```
# image: input images
# model: network
# row_idx, col_idx, patch_size: select a patch of the image to calculate Jacobian matrix

from functools import partial
import torch.nn.functional as functional

def Jacobian_rank(image, model):
    # select a patch of the image to calculate Jacobian matrix
    assert image.size(2) == image.size(3)
    image_size = image.size(2)
    image = image[:, :, row_idx:row_idx + patch_size, col_idx:col_idx + patch_size]
    zero_pad = partial(functional.pad, pad=[(image_size - patch_size) // 2 for _ in range(4)], value=0.)

    # calculate the jacobian matrix
    jacobian_matrix = jacobian(partial(net.forward, preprocess=zero_pad), image)

    # adopt trick to predict the singular values
    jacob = jacob.view(-1, image.size(1) * patch_size * patch_size)
    jacob = matmul(jacob.T, jacob)

    # calculate the partial rank of Jacobian matrix
    return matrix_rank(jacob, symmetric= True)
```

`matmul`: matrix multiplication; `jacobian`: calculate the jacobian matrix; `matrix_rank`: calculate the numerical rank of matrix.

Algorithm A2 Pseudocode of Perturbed PCA Dimension of Feature Spaces.

```
# image: input images
# model: network
# mag_perturb: magnitude of perturbations
# n_perturb: number of perturbations

def Perturbed_Dimension(image, model, mag_perturb=1e-3, n_perturb=5000):
    # extract features with random perturbations
    features = []
    for _ in range(n_perturb):
        # sample random perturbation from Gaussian distribution
        perturb = randn_like(image) * mag_perturb
        # extract feature
        feature = model(image + perturb)
        features.append(feature)
    features = concatenation(features, dim=0)

    # calculate the covariance matrix
    x = input- mean(input, dim=0)
    x = x.view(x.size(0), -1)
    cov_matrix = matmul(x.T, x) # covariance matrix

    # calculate the perturbed PCA dimensions
    return matrix_rank(cov_matrix, symmetric= True)
```

`matmul`: matrix multiplication; `randn_like`: sample a random tensor from a Gaussian distribution; `matrix_rank`: calculate the numerical rank of matrix.

Algorithm A3 Pseudocode of the Classification Dimension of the Final Feature Manifold.

```
# image: input images
# model: network
# target: ground-truth labels
# acc_ratio: threshold for measuring intrinsic dimensions of final features

def PCA(X, n_components):
    n = X.shape[0]
    X_mean = mean(X, dim=0, keepdim=True)
    X = X - X_mean
    covariance_matrix = 1 / n * matmul(X.T, X)
    eigenvalues, eigenvectors = evd(covariance_matrix, eigenvectors=True)
    eigenvalues = norm(eigenvalues, dim=1) # modulus of complex numbers
    idx = argsort(-eigenvalues)
    eigenvectors = eigenvectors[:, idx]
    eigenvectors = eigenvectors[:, :n_components]
    return eigenvectors

def Feature_projection(X, V):
    X_proj = zeros_like(X)
    for component_idx in range(V.size(1)):
        eig_vec = V[:, component_idx].unsqueeze(-1)
        eig_vec_norm = eig_vec / norm(eig_vec, p=2, keepdim=True)
        w_proj = matmul(X, eig_vec_norm)
        X_proj_i = w_proj * eig_vec_norm.T
        X_proj += X_proj_i
    return X_proj

def Intrinsic_dimension(image, model, target, acc_ratio=0.95):
    # pre-extract features and calculate original classification accuracy
    feats = model(image) # [n_samples * n_channels]
    acc_ori = calc_acc(feats, target)

    for n_component in range(1, feats.size(1)):
        # compute the eigenvalues and eigenvectors of a real square matrix
        components = PCA(feats, n_component) # [n_channels * n_component]

        # reconstruct features with principal components
        feats_rec = Feature_projection(feats, components)

        # calculate classification accuracy
        acc = calc_acc(feats_rec, target)

        # return classification dimension
        if acc >= acc_ratio * acc_ori:
            return n_component
```

`matmul`: matrix multiplication; `evd`: eigen value decomposition; `calc_acc`: calculating classification accuracy.

Algorithm A4 Pseudocode of Independence Deficit.

```
# image: input images
# model: network
# target2index: dictionary mapping from category index to sample indices
# lr: learning rate for Lasso optimization
# n_iteration: number of iterations for Lasso optimization
# w_reg: weight of the L1 regularization term

def Feature_split(feats, class_i, target2index):
    sample_indices = target2index[class_i]
    start_idx, end_idx = sample_indices[0], sample_indices[-1]
    feats_i = feats[start_idx:end_idx+1, :]
    feats_i_n = concatenation((feats_i[:, :class_i], feats_i[:, class_i+1:]), dim=1)
    feats_i_p = feats_i[:, class_i:class_i+1]
    return feats_i_n, feats_i_p

def Independence_deficit(image, model, target2index, lr=1e-5, n_iteration=5000, w_reg=20.0):
    # pre-extract logits
    logits = model(image) # [n_samples * n_classes]

    # Lasso optimization
    for class_i in range(logits.size(1)):
        # split features by category index
        feats_n, feats_p = Feature_split(logits, class_i, target2index)

        # initialize the linear coefficients of category i
        param = Parameter(zeros(feats_n.size(1), 1))

        # start training
        for _ in range(n_iteration):
            loss = mse(matmul(feat_n, param), feat_p) + w_reg * l1_norm(param)
            loss.backward()
            param -= lr * param.grad

        # save trained coefficients
        save(param)
```

`matmul`: matrix multiplication; `mse`: mean squared error; `l1_norm`: sum of the magnitudes of the vectors in a space.