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 led 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 monotone decreasing property of network ranks 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 ranks in realistic 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 [20]. 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 [46, 54, 35, 52, 47], network pruning [31, 53, 5, 24, 9], data mining [6, 23, 10, 55, 17, 28], computer vision [57, 56, 30, 26, 28, 58], and natural language processing [8, 27, 7, 11]. 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 [14] and BatchNorms [12, 4] 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 rule of differential operators and matrix multiplication, are enough to establish a universal principle—that network rank decreases monotonically in the depth of the network. Two factors further enhance the speed of decrease: 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 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 diminish 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 [18] 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 rank of its Jacobi matrix $J_{\boldsymbol{f}}$ over its input domain $\mathcal{X}$, which is defined as

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

The rank of a function represents the volume of information captured by it in the output [20]. 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 [20], we can know that rank of the function equals the intrinsic dimension of its output feature space, as captured by the following lemma.[2]

Lemma 1. *Suppose that $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^d$ is smooth almost everywhere. Let $\operatorname{Rank}(\boldsymbol{f}) = r$. 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 an open subset $\mathcal{O} \subset \mathcal{X}$, then we have $\dim(\boldsymbol{f}(\mathcal{X})) = \operatorname{Rank}(\boldsymbol{J}_{\boldsymbol{f}\circ\boldsymbol{\phi}}) \leq r$. 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 supplementary material.

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 digital accuracy of $\mathrm{float32}$, $1.19e-7$ [38], 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 [40]. 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 measure for finite sets. We can prove that the numerical rank is stable under small perturbations. Based on Weyl inequalities [48], 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 [33] 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 $\boldsymbol{J}_{\boldsymbol{f}}$, 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

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

where $1 \leq j_1 < \ldots < j_K \leq n$. 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 $\mathrm{Sub}(\boldsymbol{x}, j_1, ..., j_K) = (\boldsymbol{x}_{j_1}, ..., \boldsymbol{x}_{j_K})^T \in \mathbb{R}^K$, and $\boldsymbol{\psi}$ pad $\mathrm{Sub}(\boldsymbol{x}, j_1, ..., j_K)$ to the spatial size of $\boldsymbol{x}$ with zeros: $\boldsymbol{\psi}(\mathrm{Sub}(\boldsymbol{x}, j_1, ..., j_K)) = (0, .., 0, \boldsymbol{x}_{j_1}, 0, ..., \boldsymbol{x}_{j_K}, 0, ..., 0)^T \in \mathbb{R}^n$ with $\boldsymbol{\psi}(\mathrm{Sub}(\boldsymbol{x}, j_1, ..., j_K))_{j_k} = \boldsymbol{x}_{j_k}, k = 1, ..., K$. We then have $\boldsymbol{J}_{\boldsymbol{f}\circ\boldsymbol{\psi}} = \mathrm{Sub}(\boldsymbol{J}_{\boldsymbol{f}}, j_1, ..., j_K)$. 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 $\boldsymbol{f}_1, \boldsymbol{f}_2$, $|\mathrm{Rank}(\boldsymbol{f}_1) - \mathrm{Rank}(\boldsymbol{f}_2 \circ \boldsymbol{f}_1)| \geq |\mathrm{Rank}(\mathrm{Sub}(\boldsymbol{f}_1, j_1, \ldots, j_K)) - \mathrm{Rank}(\mathrm{Sub}(\boldsymbol{f}_2 \circ \boldsymbol{f}_1, j_1, \ldots, j_K))|, \forall 1 \leq K \leq n, 1 \leq j_1, \ldots, j_K \leq n$. Thus variance of partial ranks of adjacent sub-networks gives a lower bound on the variance of their ranks.*

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. Given network slice $\boldsymbol{F}: \mathbb{R}^n \to \mathbb{R}^d$ from input $\mathcal{X} \subset \mathbb{R}^n$ to final feature space $\boldsymbol{F}(\mathcal{X}) \subset \mathbb{R}^d$, we independently sample N points from random variable $\boldsymbol{F}(\boldsymbol{x}), \boldsymbol{x} \sim \mathbb{P}_{\mathcal{X}}$, where $\mathbb{P}_{\mathcal{X}}$ is the distribution of validation set of data $\mathcal{X}$. We then compute the covariance matrix Σ of those N samples, and eigenvectors $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^d$ of Σ, sorted by their eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$. Let $\mathrm{cls}: \mathbb{R}^d \to \mathbb{R}^c$ be the classification predictions based on the final feature representation $\boldsymbol{F}(\boldsymbol{x})$, Pro_k be projection operator in Euclidean space that projects to the linear subspace spanned by top-K eigenvectors $\boldsymbol{q}^1, ..., \boldsymbol{q}^k, k \leq d$, $\mathbb{P}_{\boldsymbol{x},\boldsymbol{y}}$ be the joint distribution of sample $\boldsymbol{x}$ and its label $\boldsymbol{y}$, 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 Principle of Rank Diminishing

We turn to the *principle of rank diminishing*. We first give a universal justification with minimum limitation on the network, so that we can safely apply this principle to many practical scenarios.

The principle of rank diminishing describes the behavior of general neural networks with almost everywhere smooth components, which exhibits the monotone decrease of network ranks and intrinsic dimensionality of feature manifolds as follows.

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

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

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

Short Argument that the Principle Should Hold Universally. Theorem 2 is ultra intrinsic for deep neural networks. It comes directly from the chain rules of differential and basic rules of matrix multiplications. The basic rule of matrix multiplication tells that, for any two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we have $\mathrm{Rank}(\boldsymbol{AB}) \leq \min\{\mathrm{Rank}(\boldsymbol{A}), \mathrm{Rank}(\boldsymbol{B})\}$ [21]. 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). Applying Lemma 1 to Eq. (7) then yields Eq. (8).

Chance of Equal Signs Holding is Small. A hypothetical but not practical concern would be that, is it possible that most of the equal signs of Eqs. (7) and (8) hold, so that the rank of network remains no significant dropping throughout the network? This concern can be mitigated by empirical and theoretical arguments. In what follows we will find that, 1) in practice, the rank of sub-networks decreases significantly after applying subsequent layers as shown in Fig. 1, and 2) in theory, there are two strong impetuses in deep neural networks to enforce strict decreasing of ranks which we will discuss in Secs. 4.1 and 4.2.

4.1 Structural Impetus of Strict Decreasing

Numerous explicit structures of the network layers can lead to a strict decrease in network ranks. Specifically, the following theorem gives a condition for the strictly greater signs to hold in the principle of rank diminishing.

Arch.	Network	Activ.	#Param.	Main Block	#Layer	Top-1 Acc.
ResNets	ResNet-18 [18]	ReLU [36]	11.7M	Bottleneck	11	69.8%
	ResNet-50 [18]	ReLU [36]	25.6M	Bottleneck	19	76.1%
MLP-like	GluMixer-24 [41]	SiLU [19]	25.0M	Mixer-Block	24	78.1%
	ResMLP-S24 [44]	GELU [19]	30.0M	Mixer-Block	24	79.4%
Transformer	ViT-T [15]	GELU [19]	5.7M	ViT-Block	13	75.5%
	Swin-T [32]	GELU [19]	29.0M	Swin-Block	18	81.3%

Table 1: Information of networks used in empirical validations. All pretrained on ImageNet.

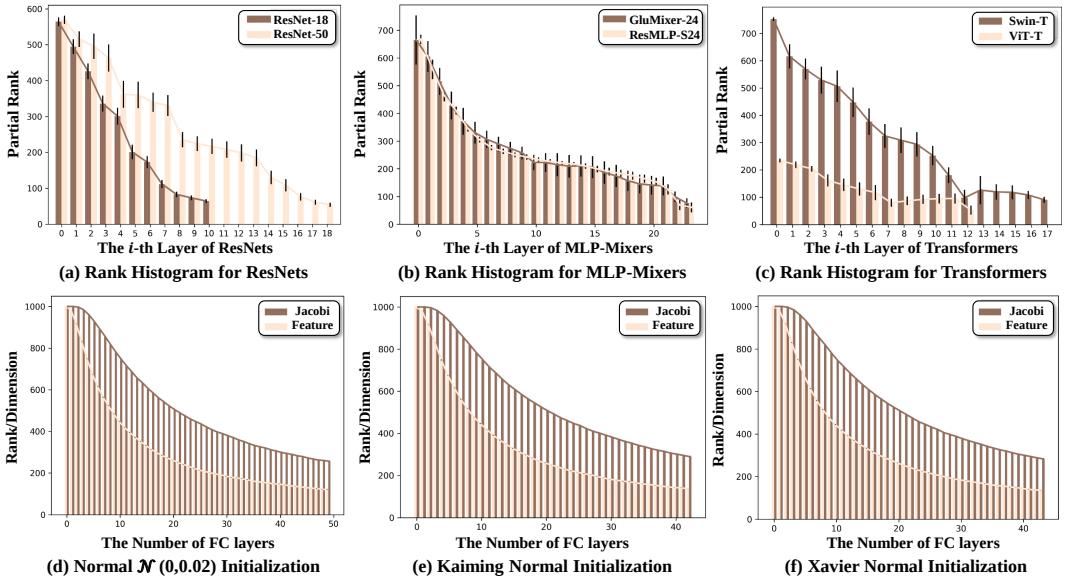


Figure 1: Partial rank of Jacobian matrices of CNN, MLP, and Transformer architecture networks for different layers on ImageNet (top row); rank of Jacobian matrices and feature dimensions of linear MLP network following conditions of Theorem 5 (bottom rule). All the models show a similar trend of exponential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

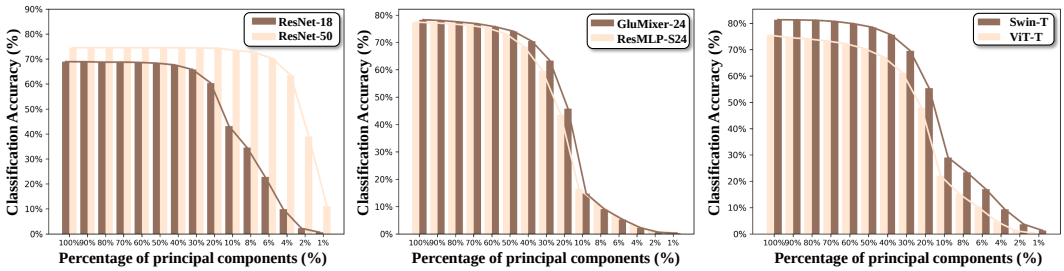
Theorem 3. [3] *Roughly speaking, if almost everywhere on the input feature manifold, there is a direction such that moving along this direction keeps the output invariant, then the intrinsic dimension of the output feature manifold will be strictly lower than that of the input. The maximum number of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

By this theorem, one can immediately find that most frequently used layer designs have high risk in inducing strict decreasing of network ranks. Normalization layers like LayerNorm [2], InstanceNorm [45], and BatchNorm [25] may lose dimensions modestly, as the output feature remains invariant along the normalized direction at each point. Linear layers like convolutions, linear transformations (e.g. dense layers), and attentions, can lose rank considerably according to the rank of their weight matrices. They constitute the explicit structural impetus to decrease network ranks and intrinsic dimensions of feature manifolds.

4.2 Implicit Impetus of Strict Decreasing

Apart from the structural impetus we propose in Sec. 4.1, there is a more intrinsic strength to pull down network ranks, which we call the implicit impetus. Deep neural networks repeatedly apply layer networks from a fixed function pool (ReLU, MLP, CNN, attention, ResNet block, *etc.*) to the input data and intermediate features to get outputs. Such paradigm accords with the cocycle dynamic systems studied by Lyapunov *et al.* [42, 50], where the Furstenberg–Kesten theorem [16] and multiplicative ergodic theorem [39] prove that logarithms of singular values divided by evolution

[3] The rigorous version is given in the supplementary material.



(a) Subspace Accuracy of ResNets (b) Subspace Accuracy of MLP-Mixers (c) Subspace Accuracy of Transformers

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. 2) 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.**

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 2: 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.

time of such chaos system converge to stable constants when time goes to infinity. While products of long chains of matrices are the simplest form of cocycle dynamic systems [29], we can get an intrinsic impetus of rank collapse tendency of Jacobian matrices independent of network architectures.

Theorem 4 (Spectrum Centralization of Function Coupling)**.** *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 denotes 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,*

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

That means for any tolerance $\epsilon > 0$, $\mathrm{Rank}_\epsilon(\boldsymbol{F})$ drops below $r + 1$ with an exponential speed as $L \to \infty$.

If further assuming Gaussian distributions of Jacobian matrices, we can prove $r = 1$ and give a more accurate estimation of the constant $\mu_r, ..., \mu_n$. As a consequence, we can find that rank of networks collapses to 1 almost surely, which is formalized in the following theorem.

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 denotes 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.

Connection with Gradient Explosion Bengio *et. al.* [3, 37] discuss the gradient explosion issue of deep neural networks, where the largest singular value of the Jacobian matrix tends to infinity when the layer gets deeper. This problem 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.

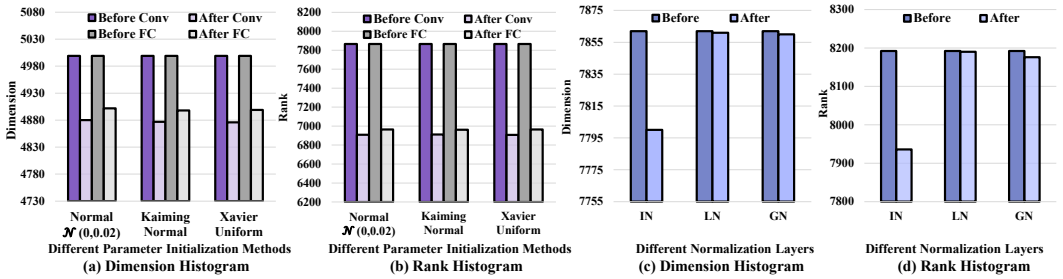


Figure 3: PCA dimension of feature spaces and rank of Jacobian matrix for commonly seen network components under standard Gaussian inputs and randomized weights. Convolution and FC layers tend to lose rank considerably; normalization layers, like InstanceNorm (IN) [45], LayerNorm (LN) [2], and GroupNorm (GN) [51], lose rank modestly. But none can preserve rank.

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

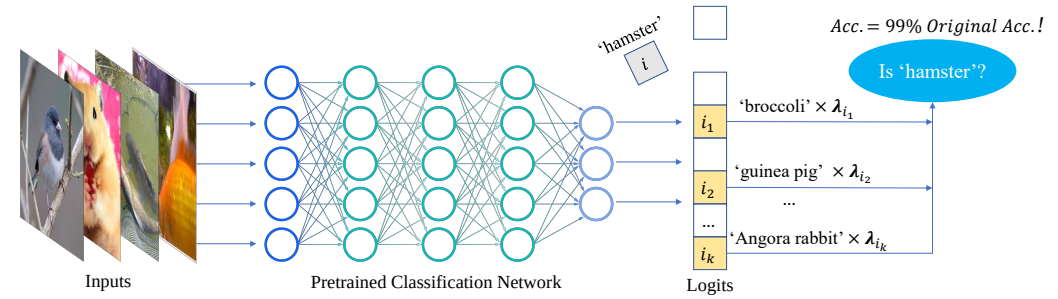
4.3 Validations

Setup In this section, we validate our theory in three types of architectures of benchmark deep neural networks, CNNs, MLPs, and Transformers, in the ImageNet [13] data domain. Information of those networks is listed in Tab. 1. 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. When measuring rank, we set $\epsilon = \text{eps} \times N$, where eps is the digital accuracy of $\text{float}32$ (*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 $\text{float}32$. All the experiments are conducted on the validation set of ImageNet and NVIDIA A100-SXM-80G GPUs.

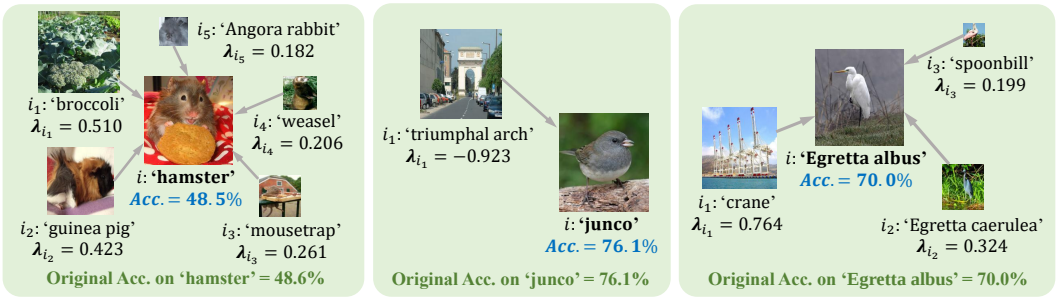
Diminishing of Rank of Jacobi As is discussed in Sec. 3.2 and Lemma 2, partial rank of Jacobian is a powerful weapon for us to detect the behavior of huge Jacobian matrices, which are infeasible to compute in practice. The decent value of partial ranks of adjacent sub-networks provides lower bound to the decent value 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. This indicates a larger rank losing in the full rank of Jacobian matrices.

Intrinsic Dimension of the Final Feature Manifold To get a further estimation of how many dimensions remain in the final feature representation, we measure the classification dimension in Fig. 2 and Tab. 2. 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. 2. 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.

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 $\text{float}32$, we stop calculation in each setting when the absolute values of elements of the matrices are lower than $1.19e-7$. We can find beautiful curves of exponential laws in all cases for both rank of Jacobian and intrinsic dimensions of features. By comparison, we can further find that the rank of benchmark deep neural networks on ImageNet bears



(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.

a striking resemblance to the exponential law of our toy setting, which confirms the proposed implicit impetus in those models.

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 [22, 49] 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 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.

Possible Remission Approaches to Rank Diminishing There are quite some techniques, at least in theory, can remiss the network rank diminishing. Typical examples are skip connection [14] and BatchNorm [12], which we will discuss in the Appendix due to page limitation.

5 Independence Deficit of Final Feature Manifolds

In this section, we provide a further perspective to study the low-rank structure of the final feature manifold, which induces an interesting finding of independence deficit in deep neural networks. We

have already known that the final feature representations of deep neural networks admit a very low intrinsic dimension. Thus there are only a few independent representations to decide the classification scores for all the 1,000 categories of ImageNet. It is then curious whether we can 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 [43],

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

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

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

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

In Fig. 4 we demonstrate 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. This means that the networks just neglect the unique representations of those categories in training and yield over-fitting when predicting them.

6 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. [14] 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. [12] 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. [34] 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 network ranks induced by the training process instead of network depth.

7 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] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

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

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

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

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

[23] 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.

[24] 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.

[25] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. Mach. Learn.*, pages 448–456. PMLR, 2015.

[26] 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.

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

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

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

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

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

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

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

[41] N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

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

[44] 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.

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

[46] 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.

[47] 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.

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

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

[50] 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.

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

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs are included in the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

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

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

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

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

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

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