THE LOW-RANK SIMPLICITY BIAS IN DEEP NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern deep neural networks are highly over-parameterized compared to the data on which they are trained, yet they often generalize remarkably well. A flurry of recent work has asked: why do deep networks not overfit to their training data? In this work, we make a series of empirical observations that investigate the hypothesis that deeper networks are inductively biased to find solutions with lower rank embeddings. We conjecture that this bias exists because the volume of functions that maps to low-rank embedding increases with depth. We show empirically that our claim holds true on finite width linear and non-linear models and show that these are the solutions that generalize well. We then show that the low-rank simplicity bias exists even after training, using a wide variety of commonly used optimizers. We found this phenomenon to be resilient to initialization, hyper-parameters, and learning methods. We further demonstrate how linear over-parameterization of deep non-linear models can be used to induce low-rank bias, improving generalization performance without changing the effective model capacity. Practically, we demonstrate that simply linearly over-parameterizing standard models at training time can improve performance on image classification tasks, including ImageNet.

1 INTRODUCTION

It has become conventional wisdom that the more layers one adds, the better a deep neural network (DNN) performs. This guideline is supported, in part, by theoretical results showing that deeper networks can require far fewer parameters than shallower networks to obtain the same modeling "capacity" (Eldan & Shamir (2016)). While it is not surprising that deeper networks are more expressive than shallower networks, the fact that state-of-the-art deep networks do not overfit, despite being heavily over-parameterized[1] , defies classical statistical theory (Geman et al. (1992); Zhang et al. (2017); Belkin et al. (2019)).

The belief that *over-parameterization via depth improves generalization* is used axiomatically in the design of neural networks. Unlike conventional regularization methods that penalize model complexity (e.g., ℓ_1/ℓ_2 penalty), over-parameterization does not. Yet, like explicit regularization, over-parameterization appears to prevent the model from over-fitting (Belkin et al. (2018); Nakkiran et al. (2019a)). Why this *implicit regularization* works is still an ongoing area of research. Even in the zero-training error regime, it is commonly observed that increasing the number of parameters improves generalization performance. Currently, the status quo explanation for this phenomenon is that gradient descent in over-parameterized models acts as a nuclear norm regularizer (Gunasekar et al. (2017); Arora et al. (2019a); Li et al. (2020)).

This work provides a new set of observations that expand the growing body of work on over-parameterization and highlights the central role of depth in finding solutions that map to low-rank embeddings for both linear and non-linear networks. Mainly, we make series of empirical observations that indicate deep networks have an inductive bias to find lower-rank embeddings. First, we observe that random deep networks are biased to map data to a feature space whose Gram matrix has a low rank. Next, we find that this low-rank phenomenon exists even after training with gradient descent. We further observe that the bias towards low-rank embeddings exists in a wide variety of common optimizers — even optimizers that do not use gradient descent. Moreover, we find that regardless of the initialization, the rank of the converged solution is largely dependent on the depth of the model. This set of observations leads us to conjecture that deeper networks are implicitly biased

[1] e.g., Dosovitskiy et al. (2020) trains a 632 million parameter, 200+ layer model, on 1.3 million images.

to find lower rank embeddings because the volume of functions that map to low-rank embeddings increases with depth. We then leverage our observations to demonstrate how one could use "depth" as a practical regularizer to achieve better generalization performance on standard benchmarks such as CIFAR (Krizhevsky et al. (2009)) and ImageNet (Russakovsky et al. (2015)).

2 PRELIMINARIES

2.1 NEURAL NETWORKS AND OVER-PARAMETERIZATION

Simple linear network A simple linear neural network transforms input $x \in \mathbb{R}^{n\times 1}$ to output $\hat{y} \in \mathbb{R}^{m\times 1}$, with a learnable parameter matrix $W \in \mathbb{R}^{m\times n}$,

$$\hat{y} = Wx. \tag{1}$$

For notational convenience, we omit the bias term.

Over-parameterized linear networks

One can over-parameterize a *linear* neural network by defining d matrices $\{W_i\}_{i=1}^{d}$ and multiplying them successively with input x:

$$\hat{y} = W_d W_{d-1} \cdots W_1 x = W_e x, \tag{2}$$

where $W_e = \prod_{i=1}^{d} W_i$. As long as the matrices are of the correct dimensionality — matrix W_d has m columns, W_1 has n rows, and all intermediate dimensions $\{\dim(W_i)\}_{i=2}^{d-1} \geq \min(m, n)$ — then this over-parameterization expresses the same set of functions as a single-layer network. We disambiguate between the collapsed and expanded set of weights by referring to $\{W_i\}$ as the over-parameterized weights and W_e as the *end-to-end* or the *effective weights*.

Non-linear networks For *non-linear* network, activation function ψ (e.g. ReLU) is interleaved in between the weights matrices:

$$\hat{y} = W_d \psi(W_{d-1} \ldots \psi(W_1(x))) \tag{3}$$

In contrast to linear networks, non-linear models become more expressive as more layers are added.

2.2 EFFECTIVE RANK

We characterize the rank of a matrix using a continuous measure known as the *effective rank*:

Definition 1 (Effective rank; Roy & Vetterli (2007))**.** *For any matrix $A \in \mathbb{R}^{m\times n}$, the effective rank ρ is defined as the Shannon entropy of the normalized singular values:*

$$\rho(A) = -\sum_{i=1}^{\min(n,m)} \bar{\sigma}_i \log(\bar{\sigma}_i),$$

where $\bar{\sigma}_i = \sigma_i / \sum_j \sigma_j$ are normalized singular values, such that $\sum_i \bar{\sigma}_i = 1$. Also referred to as the spectral entropy. Without loss of generality, we drop the exponentiation for convenience.

This measure gives us a meaningful representation of "continuous rank", which is maximized when the magnitude of the singular values are all equal and minimized when a single singular value dominates relative to others. The effective rank provides us with a metric that indicates the relative ratio of the singular values: a summarization of the distribution envelope. We use this measure primarily throughout our work, but our observations are consistent with alternative measures such as the stable-rank (Vershynin (2018)). From here on out we use *rank* and *effective rank* interchangeably.

2.3 EMBEDDING MAPS

A parameteric function $f_{\{W\}} \in \mathcal{F}_{\mathcal{W}}$ is a neural network parameterized with a set weights $\{W\} = \{W_1, \ldots, W_d\}$ that maps the input space to the output space $\mathcal{X} \to \mathcal{Y}$. For a training dataset of size q, the input and output data is $X \in \mathbb{R}^{n\times q}$ and $Y \in \mathbb{R}^{m\times q}$. Then, the predicted output is

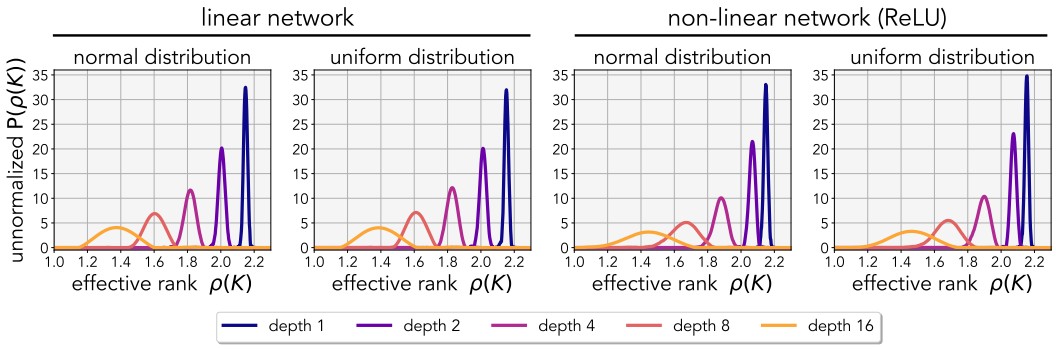


Figure 1: **Deep networks are biased toward low-rank:** The approximated probability density function (PDF) of the effective rank ρ over the Gram matrix is computed from features of the networks. The Gram matrix is computed with 256 random inputs, and we use 4096 network parameter samples to approximate the cumulative distribution function. The CDF is used to compute the PDF via the finite difference method. We apply Savitzky & Golay (1964) filter to smoothen out the approximation. There exists more probability mass for lower-rank rank embeddings when adding more layers. The experiment is repeated for both normal and uniform distributions.

$\hat{Y} = W_d \psi(\Phi) = f_{\{W\}}(X)$, where $\Phi \in \mathbb{R}^{n' \times q}$ is the last-layer embedding and $W_d \in \mathbb{R}^{m \times n'}$ is the last layer of the network.

We analyze the embedding space by computing the effective rank on the Gram/kernel matrix $K \in \mathbb{R}^{p \times p}$ where p is the size of the test set. The ij-th entry of the Gram matrix corresponds to a distance kernel $K_{ij} = \kappa(\phi_i, \phi_j)$ where ϕ_i corresponds to the i-th column of Φ. We use the model's intermediate features before the linear classifier and use cosine distance kernel [2], a common method for measuring distances in feature space (Kiros et al. (2015); Zhang et al. (2018)). Since the dimensionality of the Gram-matrix depends on the dataset size, we can compare neural networks with different modeling capacities in the zero training error regime.

Gram matrices are often used to analyze optimization and generalization properties of neural networks (Zhang et al. (2019); Du et al. (2018; 2019); Wu et al. (2019); Arora et al. (2019b)). In natural data, it is often assumed that we are trying to discover a low-rank relationship between the input and the label. For example, a model that overfits to every training sample without inferring any structure on the data will generally have a test gram-matrix that is higher rank than that of a model that has learned parsimonious representations. Lower rank on held-out data indicates less excess variability and is indicative for studying generalization and robustness. The intuition becomes clearer in linear networks, since the rank of Gram matrix depends on the rank of the linear transformation computed by the network. We illustrate this empirically in Appendix K, where we see that there is a tight relationship between the rank of the linear weight matrix and the resulting Gram matrix.

2.4 Least squares

Given a dataset X, Y generated from W^*, the goal is to regress a parameterized function $f_{\{W\}}(\cdot)$ to minimize the squared-distance $\|f_{\{W\}}(X) - Y\|_2^2$. The rank(W^*) is a measure of the "intrinsic dimensionality" of the data and we refer to it as the *task rank*. In this work, we exclusively operate in the under-determined regime where we have fewer training examples than model parameters. This ensures that there is more than one minimizing solution.

3 The inductive parameterization bias of depth

Given that our models can always fit the data, what are the implications of searching for the solution in the over-parameterized model? In linear models, this is equivalent to searching for solutions in $\{W_i\}$ versus directly in W_e. One difference is that the gradient direction $\nabla_{\{W_i\}} \mathcal{L}(\{W_i\})$ is usually different than $\nabla_{W_e} \mathcal{L}(W_e)$ for a typical loss function $\mathcal{L}$ (see Appendix H). The consequences of this

[2] cosine kernel: $\kappa(\phi_i, \phi_j) = \frac{\phi_i \phi_j^T}{\|\phi_i\| \|\phi_j\|}$

difference have been previously studied in linear models by Arora et al. (2018; 2019a), where the over-parameterized update rule has been shown to accelerate training and encourage singular values to decay faster, resulting in low nuclear-norm solution. Here we motivate a result from the perspective of parameter volume space.

Conjecture 1. *Deeper networks have a greater proportion of parameter space that maps the input data to lower-rank embeddings; hence, deeper models are more likely to converge to functions that learn simpler embeddings.*

We now provide a set of empirical observations that supports our conjecture. Our work and existing theoretical works on gradient descent biases are *not* mutually exclusive and are a likely compliment. We emphasize that we do *not* make any claims on the simplicity of the function, but only on the simplicity – lower effective rank – of the embeddings.

3.1 Low-rank simplicity bias of deep networks

Observation 1. *Randomly initialized deep neural networks are biased to correspond to a Gram matrix with a low effective rank.*

When sampling random neural networks, both linear and non-linear, we observed that the Gram matrices computed from deeper networks have lower effective rank. We quantify this observation by computing the distribution over the effective rank of the Gram matrix in Figure 1. Here, the weights of the neural networks are initialized using uniform $W_i \sim \mathcal{U}(\cdot,\cdot)$ or Normal distributions $W_i \sim \mathcal{N}(\cdot,\cdot)$. The input, output, and intermediate dimensions are 32, giving parameters $\{W_i\} \in \mathbb{R}^{d\times 32\times 32}$ for a network with d layers. We draw 4096 random parameter samples and compute the effective rank on the resulting Gram matrix. We see that the distribution density shifts towards the left (lower effective rank) when increasing the number of layers. These distributions have small overlap and smoothen out with increased depth. This observation supports the claim that depth correlates with lower effective rank embeddings.

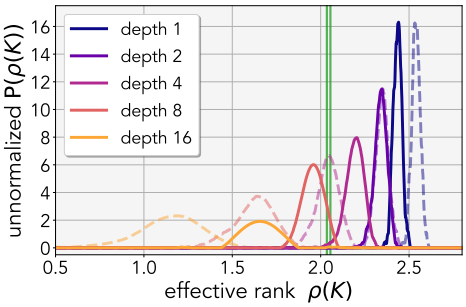


Figure 2: **Distribution at convergence**: Rank distribution after training the network with gradient descent. The dotted line indicates the initial distribution, the solid line indicates the converged distribution, and the green line indicates the ground-truth task rank.

The low-rank bias becomes more intuitive in linear models as there is a simple way to relate the Gram matrix to the weights of the model $K \approx (W_{d-1:1}X)^T(W_{d-1:1}X)$. Intuitively, if any constituent matrices are low-rank, then the product of matrices will also be low-rank – the product of matrices can only decrease the rank of the resulting matrix: $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$ (Friedberg et al. (2003)). In Appendix K, we show that as the depth of the model increases, both the effective rank of the Gram matrix and the weights decrease together. Another way to interpret our observation is that for linear models, over-parameterization does not increase the expressivity of the function but re-weights the likelihood of a subset of parameters – the hypothesis class. For non-linear models, we *cannot* make the same claims.

Although uniformly sampling under the parameter distribution is an unbiased estimator of the volume of the parameter space, it is certainly possible that a sub-space of the parameters is more likely to be observed under gradient descent. Hence, by naively sampling networks, we may never encounter model parameters that gradient descent explores. In light of this, we repeat our experiment above by computing the PDF on randomly sampled parameters after taking n gradient descent steps.

Observation 2. *Deep neural networks trained with gradient descent also learns to map data to simple embedding with low effective rank.*

Figure 2 illustrates the change in distribution as we train our model to convergence using gradient descent. Each randomly drawn network sample is trained to minimize the least-squares error. The initial distribution is plotted with dotted lines, and the converged distribution is plotted with solid lines. As the model is trained, the distribution of the rank shifts towards the ground-truth rank (green line). We visualize the learned Kernels across various non-linearities in Appendix J. Training the model with gradient descent results in a distribution that is still largely dependent on depth; this reaffirms that the role of optimization does not remove the parameterization bias deep models. In fact, if the bias stems from the model's parameterization, the same bias must also exist under other common and natural choices of optimizers. We investigate this claim below.

3.2 Is the low-rank bias specific to gradient descent?

Observation 3. *Deep neural networks trained with common and natural choices of optimizers also exhibit the low-rank embedding bias.*

The low-rank bias of deep networks has been primarily studied under the context of first-order gradient decent (Arora et al. (2018; 2019a)): *how and why does gradient descent converge to low nuclear norm solution*. In contrast, our conjecture focuses on the bias of parameterization of the network and *not* on the bias introduced by the gradient descent. Since parameterization bias exists regardless of the optimizer choice, we would expect to observe the low-rank simplicity bias on a wide range of optimizers. We directly show this in Figure 3 by ablating across various popular choices of optimizers on least-squares with linear networks. Here, we compare against Nesterov (Nesterov (1983); momentum), ADAM (Kingma & Ba (2015); hessian approximator), L-BFGS (Liu & Nocedal (1989); second-order), CMA-ES (Hansen et al. (2003) evolutionary-search), and random search. All models were trained to zero training error except for random-search. For random search, we randomly initialize the network $100,000$ times and take the best performing sample. As we have seen with gradient descent, the experiment indicates that even when we train with a wide suite of commonly used optimizers, the solution obtained by these models depends on how the model was originally parameterized.

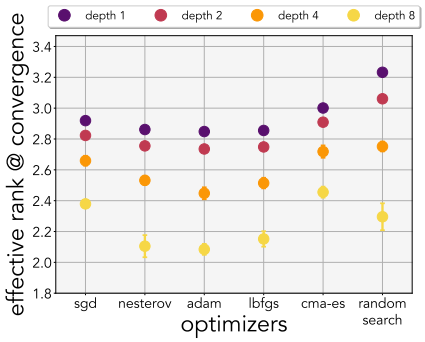


Figure 3: **Low-rank bias & optimizers**: Least-squares trained on linear neural networks using various optimization methods. The rank of the converged Gram matrix is correlated with the depth of the network. The experiment is repeated 5 times.

3.3 Can the simplicity bias be explained solely by the initialization?

The previous set of experiments indicate that deeper networks are biased towards low-rank embedding at both initialization and convergence. In these experiments, the random settings of neural networks had different initial distributions. This happens because, even if the individual weights are normally distributed, the weights constructed from a series of matrix multiplications result in a distribution that has a high density around zero. For example, the product of 2 normally distributed weights becomes symmetric χ-squared distribution, with 1 degrees of freedom. Hence, one could argue that the converged solutions are low-rank because of the initialization bias.

Observation 4. *Deep neural networks are biased towards learning low-rank embeddings regardless of the initialization.*

To test whether the initialization of the model affects the rank of the converged solution. We optimize our network $W \in \mathbb{R}^{d\times 32\times 32}$ on least-squares where the task-rank is set to 24. All models are trained for 4000 epochs, using gradient descent, and the best learning rate is chosen for each depth. In Figure 4 (left), for models using *default initialization*, we show that increasing the number of layers decreases the effective rank of the Gram matrix at convergence. We repeat the experiment in Figure 4 (right) by initializing the over-parameterized models with the distribution associated with the 32-layer linear network. Following a similar trend to that of default initialization, we observe that

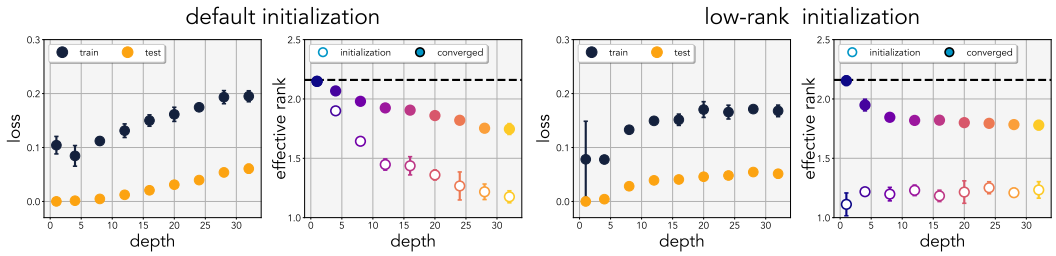


Figure 4: **Bias of parameterization:** The effective rank of the Gram matrix from initialization to convergence on various depth. For each depth, we train a linear network using gradient descent on least squares regression. We repeat our experiments 5 times with different seeds, and we report the median of these runs. The rank at initialization and convergence is indicated by white and colored dots, respectively. For deeper models, the effective-rank is lower at initialization because the product of normally distributed weights is no longer normally distributed. On the right, we initialize the networks with the same low-rank distribution of weights as the 32-layer model. We observe that shallower networks tend to converge to higher rank embeddings.

deeper models learn embeddings that are a lower rank than the shallower counterparts. Although initialization is not insignificant, we see that the depth of the model has tight control over the solution in which the model explores. For deeper networks, the majority of the parameter volume is mapped to low-rank embedding (Observation 1), and therefore it is expected that a typical search algorithm would likely encounter parameters that map to low-rank embeddings regardless of initialization. Similarly, for a shallower network, it would be easier to find a solution that maps to higher rank embeddings.

3.4 Relation to random matrix theory

In linear models, we have a special case in which the low-rank embedding corresponds to low-rank weights. This enables us to make a natural connection to existing theoretical work from random matrix theory (RMT), which studies the spectral distribution under matrix multilpications (Akemann et al. (2013a;b); Burda et al. (2010)). We leverage the results from Pennington et al. (2017); Neuschel (2014) to show the following:

Theorem 1. *Let ρ be the effective rank measure defined in Definition 1. For a linear neural network with d-layers, where the parameters are drawn from the same Normal distribution $\{W_i\}_{i=1}^{d} \sim \mathcal{W}$, the effective rank of the weights monotonically decreases when increasing the number of layers when $dim(W) \to \infty$,*

$$\rho\left(W_d W_{d-1} \dots W_1\right) \leq \rho\left(W_{d-1} \dots W_1\right)$$

Proof. See Appendix G.

The theory is preliminary and can only be understood in the context of an infinite width model with random matrices. Yet, we have found in practice that the empirical spectral distribution of finite-width models is well approximated by theory (see Appendix A). The main contribution of our work is on the empirical theory of the low-rank bias of deep networks; nonetheless, we show that there is a natural theoretical connection to RMT to stimulate future research.

4 Over-parameterization as a regularizer

Thus far, we have observed that depth acts as a bias for finding functions with low-rank embeddings. As one could imagine, this inductive bias of depth could be used to help but also hurt generalization performance. Our observations indicate that the low-rank simplicity bias helps when the true function we are trying to approximate is low-rank. On the contrary, if the underlying mapping is a high-rank, or the network is made too deep, depth could have a converse effect on generalization. Ample evidence from prior works (Szegedy et al. (2015); He et al. (2016)) suggests that over-parameterization of

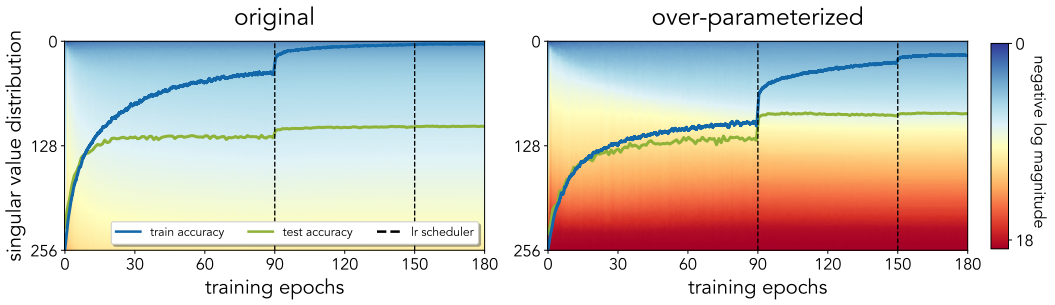


Figure 5: **Rank training dynamics:** Singular values of the Gram matrix for both original (left) and linearly over-parameterized (right) model throughout training. The models are trained on `CIFAR100` using SGD. We visualize the negative log magnitude of the normalized singular values. The dotted lines (−−) indicate the learning step schedule, and train and test losses are overlayed on top of the distribution. The over-parameterized model learns lower rank embedding and exhibits less overfitting, and has better generalization.

non-linear models improves generalization on fixed datasets, but blindly increasing the number of layers without bells & whistles (e.g., batch-norm, residual connection, etc.) hurts (He et al. (2016)).

Fortunately, networks are trained on natural data, where often the goal is to discover a low-rank relationship between the input and the label. Hence, the inductive bias of depth acts as a prior rather than a bug. As noted by Solomonoff (1964) theory of inductive inference, the simplest solution is often the best solution, suggesting that low-rank mapping in neural networks can be used to improve generalization and robustness to overfitting. However, increasing the number of non-linear layers also increases the modeling capacity, thereby making it difficult to isolate the effect of depth.

Nevertheless, since a non-linear network is composed of many linear components, such as fully connected and convolutional layers, we can over-parameterize these linear layers to induce a low-rank bias in the model without increasing the modeling capacity. The details of our linear over-parameterization method are in Appendix B. We observe that such linear over-parameterization improves generalization performance on classification tasks. Furthermore, we find that such implicit regularization outperforms models trained with several choices of explicit regularization. Guo et al. (2020) made a similar empirical observation in the context of model compression where linear over-parameterization improves generalization, but why it works is unexplored.

4.1 Image classification with over-parameterization

Using the linear expansion rules in Appendix B, we over-parameterize various architectures and evaluate on a suite of standard image classification datasets: `CIFAR10`, `CIFAR100`, `ImageNet`. All models are trained using SGD with a momentum of 0.9. For data augmentation, we apply a random horizontal flip and random-resized crop. We follow standard training procedures[3] and only modify the network architecture.

Expansion			CIFAR10		CIFAR100	
Factor	FC	Conv	accuracy	gain ↑	accuracy	gain ↑
1x	-	-	86.9	-	57.0	-
2x	✓	-	87.1	+0.2	58.4	+1.4
2x	-	✓	87.8	+0.9	61.0	+4.0
2x	✓	✓	**89.1**	**+2.2**	61.2	+4.2
4x	✓	-	87.3	+0.4	59.7	+2.7
4x	-	✓	**89.1**	**+2.2**	61.3	+4.3
4x	✓	✓	89.0	+2.1	**63.5**	**+6.5**
8x	✓	-	85.9	-1.0	58.8	+1.8
8x	-	✓	88.5	+1.6	61.6	+4.6
8x	✓	✓	88.0	+1.1	61.5	+4.5

Table 1: **Over-parameterization ablations:** A nonlinear CNN with 4 convolution and 2 linear layers trained on `CIFAR10` and `CIFAR100` with various degrees of linear over-parameterization.

In Figure 5, we compare a CNN trained without (left) and with (right) our over-parameterization (expansion factor $d = 4$) on `CIFAR100`. The CNN consists of 4 convolutional layers and 2 fully connected layers; the architecture details are in Appendix E. We overlay the dynamics of the singular values of the Gram matrix throughout training. The spectral distribution is normalized by the largest singular value and are sorted in descending order $\sigma_i(A) \geq \sigma_{i+1}(A)$ for $i < 1 \leq \min(m, n)$. We observe that both the individual effective weights and the Gram matrix of the over-parameterized model is biased towards low-rank weights. Unlike the original, the majority of the singular values of the over-parameterized model are

[3] https://github.com/pytorch/examples/blob/master/imagenet

close to zero. Both original and linearly over-parameterized models first exhibit rank contracting behavior throughout training, and then the rank starts to increase again – to the best of our knowledge, this is an unexpected training behavior in larger models that are not explained in prior works.

We further quantify the gain in performance from linear over-parameterization in Table 1. The learning rate is tuned per configuration, and we report the best test accuracy throughout the training. We try various over-parameterization configurations and find an expansion factor of 4 to be the sweet spot, with a gain of $+6.3$ for `CIFAR100` and $+2.8$ for `CIFAR10`. The optimal expansion factor depends on the depth of the original network, and in general, we observe a consistent improvement for over-parameterizing models with < 20 layers on image classification.

architecture	**ImageNet**		
	original	over-param	gain ↑
AlexNet (2x)	57.3	59.1	**+1.8**
ResNet10 (2x)	62.8	63.7	**+0.9**
ResNet18 (2x)	67.3	67.7	**+0.4**

Table 2: **ImageNet:** We show on existing architectures that linear over-parameterization can improve generalization performance. The benefit plateaus off when using deeper models. We did not see a noticeable improvement starting from `ResNet34`.

We scale up our experiments to ImageNet, a large-scale dataset consisting of 1.3 million images with 1000 classes, and show that our findings hold in practical settings. For these experiments, we use standardized architectures: AlexNet (Krizhevsky et al. (2012)) which consists of 8-layers, and ResNet10 / ResNet18 (He et al. (2016)) which consists of 10 and 18 layers, respectively. If our prior observations hold true, we would expect the gain in performance from over-parameterization to be reduced for deeper models. This is, in fact, what we observed in Table 2, with moderate gains in AlexNet and less for ResNet10 and even less for ResNet18. In fact, starting from ResNet34, we observe linearly over-parameterized models perform worse than the original. These experiments support our claim that adding too many layers can over-penalize the model.

To find out whether explicit regularizers can approximate the advantages of over-parameterization, we directly compare the performance in Table 3 on `CIFAR`. These regularizers include popular ℓ_1 and ℓ_2 norm-based regularizers and commonly-used pseudo-measures of rank. These pseudo-measures of rank, such as *effective rank* and *nuclear norm*, require one to compute the singular value decomposition, which is computationally infeasible on large-scale models. Although we found explicit rank regularizers to help, we observed over-parameterization to out-perform models trained with explicit regularizers. Moreover, we found that combining norm-based regularizers with over-parameterization further improves performance. This discrepancy between implicit and explicit regularization may stem from the fact that over-parameterization receives a combined effect of both gradient descent's implicit bias and model parameterization's inductive bias. Therefore, one may need to jointly consider both biases to approximate its effect as an explicit regularizer correctly. Another reason could be that regularizers are inherently different than over-parameterization (Arora et al. (2018)). For example, a model trained with a regularizer will have a non-zero gradient, even at zero training loss, while the over-parameterized model will not.

regularization	**CIFAR10**		**CIFAR100**	
	accuracy	gain ↑	accuracy	gain ↑
none (baseline)	86.9	-	57.0	-
low-rank initialization	86.8	-0.1	57.2	+0.2
ℓ_2 norm	87.2	+0.3	57.0	+0.0
ℓ_1 norm	87.4	+0.5	60.0	+3.0
nuclear norm	87.0	+0.1	58.1	+1.1
effective rank	86.9	+0.0	57.2	+0.2
stable rank	87.6	+0.9	58.3	+1.3
frobenius^2 norm	87.0	+0.1	59.2	+2.2
over-param (2x)	89.1	+2.2	61.2	+4.2
over-param (2x) + ℓ_2	89.6	+2.7	61.1	+4.1
over-param (2x) + ℓ_1	**89.7**	**+2.8**	**63.3**	**+6.3**

Table 3: **Explicit regularizers:** Comparison of models trained with various regularizers. Over-parameterization consistently outperforms explicit regularizers.

5 Discussion

We conclude by discussing several implications of our work:

Parameterization matters One of the main ingredients in any machine learning algorithm is the choice of hypothesis space: what is the set of functions under consideration for fitting the data? Although this is a critical choice, *how the hypothesis space is also parameterized matters*. Even if two models span the same hypothesis space, the way we parameterize the hypothesis space can ultimately determine which solution the model will converge to – recent work has shown that networks with the better neural reparameterizations can find more effective solutions (Hoyer et al. (2019)).

The automation of finding the right parameterization also has a relationship to neural architecture search (Zoph & Le (2017)), but architecture search typically conflates the search for better hypothesis spaces with the search for better parameterizations of given hypothesis space. In this work, we have explored just one way of reparameterizing neural nets – stacking linear layers – which does not change the hypothesis space, but many other options exist (see Figure 7 and a short extension to residual networks Appendix C). Understanding the biases induced by these reparameterizations may yield benefits in model analysis and architectural designs.

Deep over-parameterization vs. regularization We have argued that depth acts as an inductive bias toward simple (low-rank) solutions because the relative volume of simple solutions in deeper parameter space is larger, extending prior arguments by Valle-Perez et al. (2019). This effect is qualitatively different than explicit regularization in that it does not change the objective being minimized. Occam's razor states that the best solution is the simplest that fits the data (as has been formalized in various notions of optimal inference (Solomonoff (1964))). Too strong an explicit regularizer can prefer simple solutions that do not fit the data, which is incorrect when the true solution is, in fact, complex. Picking a parameterization in which simple solutions are more prevalent but not enforced is an alternative way to approximate the idea of Occam's razor. Our experiments, which show that explicit rank regularization underperforms compared to deep over-parameterization, suggest that this implicit regularization may have advantages.

6 Related works

Linear networks Linear networks have been used in lieu of non-linear networks for analyzing the generalization capabilities of deep networks. These networks have been widely used for analyzing learning dynamics (Saxe et al. (2014)) and forming generalization bounds (Advani et al. (2020)). Notable work from Arora et al. (2018) shows that over-parameterization induces training acceleration which cannot be approximated by an explicit regularizer. Furthermore, Gunasekar et al. (2017) shows that linear models with gradient descent converge to a minimum nuclear norm solution on matrix factorization. More recently, Li et al. (2020) demonstrated gradient descent acts as a greedy rank minimizer in matrix factorization, and Bartlett et al. (2020; 2021) argues that gradient descent in over-parameterized models lead to benign overfitting. Although mainly used for simplifying theory, Bell-Kligler et al. (2019) demonstrate the practical applications of deep linear networks.

Low-rank bias Deep linear neural networks have been known to be biased towards low-rank solutions. One of the most widely studied regime is on matrix factorization with gradient descent under isometric assumptions (Tu et al. (2016); Ma et al. (2018); Li et al. (2018)), and further studied on least-squares (Gidel et al. (2019)). Arora et al. (2019a) showed that matrix factorization tends to low nuclear-norm solutions with singular values decaying faster in deeper networks. Note that these work focuses on why gradient descent finds low-rank solutions. Pennington et al. (2018) showed that the input-output Jacobian's spectral distribution is determined by depth. For non-linear networks, understanding the biases has been mostly empirical, with the common theme that over-parameterization of depth or width improves generalization (Neyshabur et al. (2015); Nichani et al. (2020); Golubeva et al. (2021); Hestness et al. (2017); Kaplan et al. (2020)). These aforementioned theories have also been adopted for auto-encoding Jing et al. (2020) and model compression, Guo et al. (2020). More recently, Pezeshki et al. (2020) have observed that SGD learns to capture statistically dominant features which leads to learning low-rank solutions, and Baratin et al. (2021) observed that the alignment of the features act as an implicit regularizer during training.

Simplicity bias Recent work has indicated that gradient descent in linear models find max-margin solutions Soudry et al. (2018); Nacson et al. (2019); Gunasekar et al. (2018). Separately, in the perspective of algorithmic information theory, Valle-Perez et al. (2019) demonstrated that deep networks' parameter space maps to low-complexity functions. Furthermore, Nakkiran et al. (2019b) and Arpit et al. (2017) have shown that networks learn in stages of increasing complexity. Whether these aspects of simplicity bias are desireable has been studied by Shah et al. (2020).

Complexity measures A growing number of work has found matrix norm to not be a good measure for characterizing neural networks. Shah et al. (2018) shows that minimum norm solution is not guaranteed to generalize well. These findings are echoed by Razin & Cohen (2020), which demonstrates that implicit regularization cannot be characterized by norms and proposes rank as an alternative measure.

REFERENCES

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

Gernot Akemann, Jesper R Ipsen, and Mario Kieburg. Products of rectangular random matrices: singular values and progressive scattering. *Physical Review E*, 88(5):052118, 2013a.

Gernot Akemann, Mario Kieburg, and Lu Wei. Singular value correlation functions for products of wishart random matrices. *Journal of Physics A: Mathematical and Theoretical*, 46(27):275205, 2013b.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *ICML*, 2018.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019a.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pp. 322–332. PMLR, 2019b.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pp. 2269–2277. PMLR, 2021.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *arXiv preprint arXiv:2103.09177*, 2021.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *Advances in Neural Information Processing Systems*, 2019.

Zdzisław Burda, Andrzej Jarosz, Giacomo Livan, Maciej A Nowak, and Artur Swiech. Eigenvalues and singular values of products of rectangular gaussian random matrices. *Physical Review E*, 82 (6):061114, 2010.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pp. 1675–1685. PMLR, 2019.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pp. 907–940. PMLR, 2016.

S.H. Friedberg, A.J. Insel, and L.E. Spence. *Linear Algebra*. Featured Titles for Linear Algebra (Advanced) Series. Pearson Education, 2003. ISBN 9780130084514. URL https://books.google.com/books?id=HCUlAQAAIAAJ.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In *Advances in Neural Information Processing Systems*, pp. 3202–3211, 2019.

Anna Golubeva, Behnam Neyshabur, and Guy Gur-Ari. Are wider nets better given the same number of parameters? In *International Conference on Learning Representations*, 2021.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*, 2015.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 6151–6159, 2017.

Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9461–9471, 2018.

Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks. In *Advances in Neural Information Processing Systems*, 2020.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. Neural reparameterization improves structural optimization. *arXiv preprint arXiv:1909.04240*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Li Jing, Jure Zbontar, et al. Implicit rank-minimizing autoencoder. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pp. 2–47. PMLR, 2018.

Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *arXiv preprint arXiv:2012.09839*, 2020.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion. In *International Conference on Machine Learning*, pp. 3345–3354. PMLR, 2018.

RV Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry. Convergence of gradient descent on separable data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3420–3428. PMLR, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019a.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *arXiv preprint arXiv:1905.11604*, 2019b.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pp. 543–547, 1983.

Thorsten Neuschel. Plancherel–rotach formulae for average characteristic polynomials of products of ginibre random matrices and the fuss–catalan distribution. *Random Matrices: Theory and Applications*, 3(01):1450003, 2014.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *International conference on machine learning*, 2015.

Eshaan Nichani, Adityanarayanan Radhakrishnan, and Caroline Uhler. Do deeper convolutional networks perform better? *arXiv preprint arXiv:2010.09610*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Jeffrey Pennington, Samuel S Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, 2017.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1924–1932. PMLR, 2018.

Mohammad Pezeshki, Sekou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *arXiv preprint arXiv:2011.09468*, 2020.

Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. In *Advances in neural information processing systems*, 2020.

Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pp. 321–352. Springer, 2005.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pp. 606–610. IEEE, 2007.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 2015.

Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

Andrew M Saxe, James L Mcclelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*. Citeseer, 2014.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *arXiv preprint arXiv:2006.07710*, 2020.

Vatsal Shah, Anastasios Kyrillidis, and Sujay Sanghavi. Minimum norm solutions do not always generalize well for over-parameterized problems. In *stat*, volume 1050, pp. 16, 2018.

Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22, 1964.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1): 2822–2878, 2018.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Stephen Tu, Ross Boczar, Max Simchowitz, Mahdi Soltanolkotabi, and Ben Recht. Low-rank solutions of linear matrix equations via procrustes flow. In *International Conference on Machine Learning*, pp. 964–973. PMLR, 2016.

Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Xiaoxia Wu, Simon S Du, and Rachel Ward. Global convergence of adaptive gradient methods for an over-parameterized neural network. *arXiv preprint arXiv:1902.07111*, 2019.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

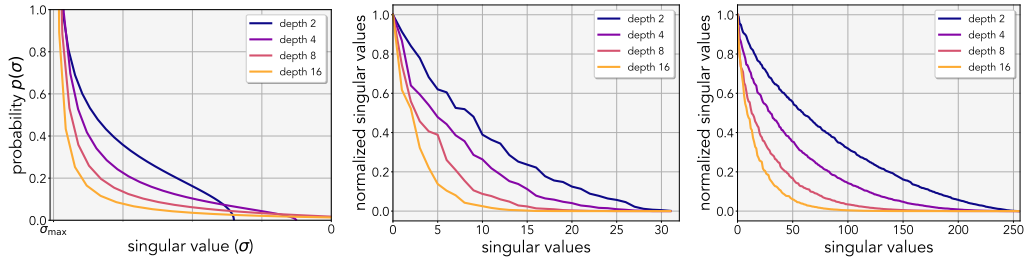
Guodong Zhang, James Martens, and Roger B Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems*, pp. 8082–8093, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

APPENDIX

A RANDOM MATRIX THEORY IN FINITE MODELS



(a) `Theoretical` Pennington et al. (2017); Neuschel (2014) (b) `Empirical` $W \in \mathbb{R}^{32\times 32}$ (c) `Empirical` $W \in \mathbb{R}^{256\times 256}$

Figure 6: **Theoretical and empirical singular-value distributions:** We show that even on finite matrices, the singular-value distribution matches that of the theoretical distribution. This implies that deeper finite-width linear neural networks should have lower effective rank in practice.

Random matrix theory makes an infinitely large matrix assumption to derive a deterministic spectral distribution of random matrices. This assumption is equivalent to making an infinite width neural network assumption. In Figure 6, we show that the empirical singular-value distribution closely follows that of the theoretical distribution. Even when using a very small weight matrix of size $W \in \mathbb{R}^{32\times 32}$, and more so on larger weight matrices $W \in \mathbb{R}^{256\times 256}$, the singular values sharpen when increasing the number of layers – reaffirming our Theorem 1 and our conjecture.

B EXPANDING A NON-LINEAR NETWORK

A deep non-linear neural network with l layers is parameterized by a set of l weights $W = \{W_1, \ldots, W_l\}$. The output of the j-th layer is defined as $\phi_j = \psi(f_{W_j}(\phi_{j-1}))$, for some non-linear function ψ and input feature ϕ_{j-1}. The initial feature map is the input $\phi_0 = x$, and the output is the final feature map $y = \phi_l$. We can expand a model by depth d by expanding all linear layers, i.e. redefining $f_{W_j} \rightarrow f_{W_j^d} \circ \cdots \circ f_{W_j^1} \; \forall \; j \in \{1, ..., l\}$. We illustrate this in Figure 7. We describe this operation for fully connected and convolutional layers.

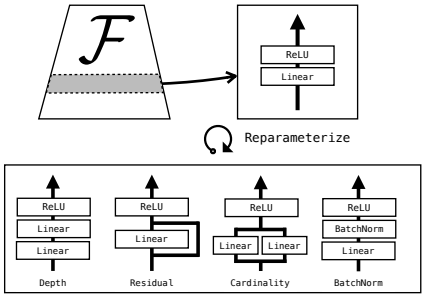


Figure 7: **Linear reparameterization**: For a model $\mathcal{F}$, we can reparameterize any linear layer to another functionally equivalent layer (shown in the box below). In this work we mainly explore reparameterization of depth. Batch-norm and any other running-statistics driven normalization layers are linear only at test time.

Fully-connected layer A fully-connected layer is parameterized by weight $W \in \mathbb{R}^{m \times n}$. One can over-parameterize W as a series of linear operators defined as $\prod_{i=1}^{d} W_i$. For example, when $d = 2$, $W \rightarrow W_2 W_1$, where $W_2 \in \mathbb{R}^{m \times h}$ and $W_1 \in \mathbb{R}^{h \times n}$ for some hidden dimension h. The variable h is referred to as the width of the expansion and can be arbitrarily chosen. In our experiments, we choose $h = n$ unless stated otherwise. Note that $h < \min(m, n)$ would result in a rank bottleneck and explicitly reduce the underlying rank of the network.

Convolutional layer A convolutional layer is parameterized by weight $W \in \mathbb{R}^{m \times n \times k \times k}$, where m and n are the output and input channels, respectively, and k is the dimensionality of the convolution kernel. For convenience, we over-parameterize by adding 1×1 convolution operations. $W_d * W_{d-1} * \cdots * W_1$, where $W_d \in \mathbb{R}^{m \times h \times 1 \times 1}$, $W_{d-1}, ..., W_2 \in \mathbb{R}^{h \times h \times 1 \times 1}$ and $W_1 \in \mathbb{R}^{h \times n \times k \times k}$. Analogous to the fully-connected layer, we choose $h = n$ to avoid rank bottleneck.

The work by Golubeva et al. (2021) explores the impact of width h. Similar to their findings, we observed using the larger expansion width to slightly improve performance. We use $h = 2n$ for our ImageNet experiments.

C EXTENSION TO RESIDUAL CONNECTIONS

This work concentrates our analysis on depth and its role in both linear and non-linear networks. Yet, the ingredients that make up what we know as state-of-the-art models today are more than just depth. From cardinality Xie et al. (2017) to normalization Ioffe & Szegedy (2015) and residual connections He et al. (2016), numerous facets of parameterization have become a fundamental recipe for a successful model (see Figure 7). Of these, residual connections have the closest relevance to our work.

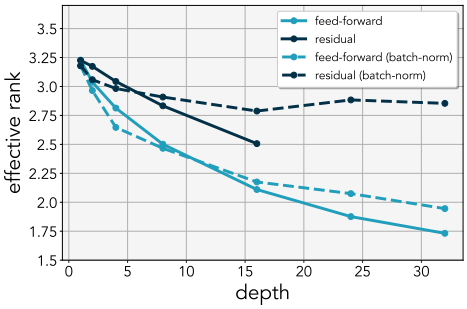


Figure 8: **Residual connections:** The effective rank of linear models trained with and without residual connection on a low-rank least-squares problem. Contrary to feed-forward networks, residual networks maintains the effective rank of the weights even when adding more layers. Residual networks without batch-normalization suffer from unstable output variance after 16 layers.

What is it about residual connections that allow the model to scale arbitrarily in depth? while vanilla feed-forward networks cannot? One possibility is that beyond a certain depth, the rank of the solution space reduces so much that good solutions no longer exist. In other words, the implicit rank-regularization of depth may take priority over the fit to training data. Residual connections are essentially "skip connections" that can be expressed as $W \to W + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix (Dirac tensor for convolutions). There are two interpretations of what these connections do: one is that identity preservation prevents the rank-collapse of the solution space. The other interpretation is that residual connections reduce the *effective depth* — the number of linear operators from the input to the output (e.g., ResNet50 and ResNet101 have the same effective depth), which prevents rank-collapse of the solution space. Results in Figure 8 confirm this intuition. ResNets, unlike linear networks, *do not* exhibit a monotonic rank contracting behavior and the effective rank plateaus after 8 layers, regardless of using batch-normalization or not. Furthermore, preliminary experiments on least-squares using linear residual networks indicate that the effective rank of the solution space is also bounded by the number of layers in the shortest and longest path from the inputs to the outputs. A thorough study on the relationship between residual connections and rank is left for future work.

D COMPARISONS OF RANK MEASURES

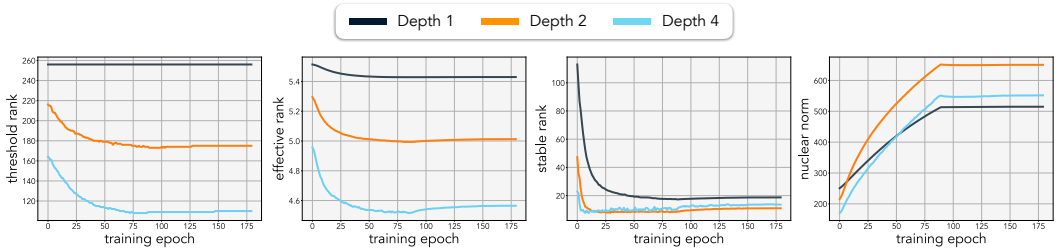


Figure 9: **Comparing rank-measures:** Comparison between various pseudo-metrics of rank when varying the number of layers. The threshold is set to $\tau = 0.01$ for threshold rank.

The rank of a matrix – which defines the number of independent basis – can often be itself a sub-optimal measure. For deep learning, fluctuations in stochastic gradient descent and numerical imprecision can easily introduce noise that causes a matrix to be full-rank. In addition, simply counting the number of non-zero singular values may not be indicative of what we care about in practice: the relative impact of the i-th basis compared to the j-th basis. Effective rank was proposed to deal with these problems, and we have extensively used them throughout our work.

Definition 2 (Effective rank)**.** *Roy & Vetterli (2007)*

For any matrix $A \in \mathbb{R}^{m \times n}$, the effective rank ρ is defined as the Shannon entropy of the normalized singular values:

$$\rho(A) = -\sum_{i=1}^{\min(n,m)} \bar{\sigma}_i \log(\bar{\sigma}_i),$$

where $\bar{\sigma}_i = \sigma_i / \sum_j \sigma_j$ are the normalized singular values such that $\sum_i \bar{\sigma}_i = 1$. It follows that $\rho(A) \leq rank(A)$. This measure is also known as the spectral entropy.

We now state other various metrics that have been used as a pseudo-measure of matrix rank. One obvious alternative is to use the original definition of rank after normalization:

Definition 3 (Threshold rank)**.** *For any matrix $A \in \mathbb{R}^{m \times n}$, the threshold rank τ-Rank is the count of non-small singular values after normalization:*

$$\tau\text{-}Rank(A) = \sum_{i=1}^{\min(n,m)} \mathbb{1}[\bar{\sigma}_i \geq \tau],$$

where $\mathbb{1}$ is the indicator function, and $\tau \in [0, 1)$ is the threshold value. $\bar{\sigma}_i$ are the normalized singular values defined above.

It is worth noting that not normalizing the singular values results in the numerical definition of rank. Although the threshold rank is the closest to the original definition of rank, depending on the threshold value, a drastically different scalar representation of rank can emerge. Potentially, better usage of threshold rank is to measure the AUC when varying the threshold.

Related to the definition of the threshold rank, stable rank operates on the normalized squared-singular values:

Definition 4 (Stable rank)**.** *Vershynin (2018)*

For any matrix, $A \in \mathbb{R}^{m \times n}$, the stable rank is defined as:

$$SRank(A) = \frac{\|A\|_F^2}{\|A\|_2} = \frac{\sum \sigma_i^2}{\sigma_{max}^2},$$

Where σ_i are the singular values of A.

Stable-rank provides the benefit of being efficient to approximate via the power iteration Mises & Pollaczek-Geiringer (1929). In general, stable-rank is a good proxy for measuring the rank of the matrix.

Lastly, the nuclear norm has been considered as the de facto measure of rank in the matrix factorization/completion community, with low nuclear-norm indicating that the matrix is low-rank:

Definition 5 (Nuclear norm)**.** *For any matrix $A \in \mathbb{R}^{m \times n}$, the nuclear norm operator is defined as:*

$$\|A\|_* = \mathrm{tr}(\sqrt{AA^T}) = \sum_{i}^{\min(n,m)} \sigma_i(A)$$

Where σ_i are the singular values of A.

Nuclear norm, however, has obvious flaws of being an un-normalized measure. The nuclear norm is dictated by the magnitude of the singular values and not the ratios. Therefore, the nuclear norm can be made arbitrarily large or small without changing the output distribution.

The comparisons of these metrics are illustrated in Figure 9. The metrics are computed on the end-to-end weights throughout the training. We use linear over-parameterized models with various depths on least-squares.

E TRAINING DETAILS AND MODEL ARCHITECTURE

All models for image classification are trained using `PyTorch` Paszke et al. (2019) with `RTX 2080Ti` GPUs. We use stochastic gradient descent with a momentum of 0.9. For CIFAR experiments, the initial learning rate is individually tuned (0.02 for most cases), and we train the model for 180 epochs. We use a step learning rate scheduler at epoch 90 and 150, decreasing the learning rate by a factor of 10 each step. For all the models, we use random-horizontal-flip and random-resize-crop for data augmentation. The training details for ImageNet can be found in `https://github.com/pytorch/examples/blob/master/imagenet`. When linearly over-parameterizing our models, we bound the variance of the weights using Kaiming initialization He et al. (2016), a scaled Normal distribution. This allows us to have the same output variance, regardless of the number of layers we over-parameterize our models by. We found this to be critical for stabilizing our training. We also found it important to re-tune the weight decay for larger models on ImageNet. The architecture used for the CIFAR experiments is:

CIFAR architecture
RGB image $y \in \mathbb{R}^{32\times32\times3}$
Convolution $3 \to 64$, MaxPool, ReLU
Convolution $64 \to 128$, MaxPool, ReLU
Convolution $128 \to 256$, MaxPool, ReLU
Convolution $256 \to 512$, ReLU
GlobalMaxPool
Fully-Connected $512 \to 256$, ReLU
Fully-Connected $256 \to$ num classes

F DIFFERENTIAL EFFECTIVE RANK

To analyze the effective rank as a function of the number of layers, we define a differential variant of the effective rank. This formulation allows us to use the fact that the eigen/singular-value spectrum assumes a probability distribution in the asymptotic case.

Definition 6 (Differential effective rank)**.** *For any matrix $A \in \mathbb{R}^{m\times n}$ as $\min(m,n) \to \infty$ the singular values assume a probability distribution $p(\sigma)$. Then, we define the differential effective rank ρ as:*

$$\rho(A) = -\int_{\sigma} \frac{\sigma}{c} \log(\frac{\sigma}{c}) p(\sigma) d\sigma \tag{4}$$

where $p(\sigma)$ is the singular value density function and $c = \int \sigma p(\sigma) d\sigma$ is the normalization constant.

G PROOF OF THEOREM 1

To prove Theorem 1, we leverage the findings from random matrix theory, where the singular values assume a probability density function. Specifically, we use the density function corresponding to the singular values of the matrix W composed of the product of L individual matrices $W = W_L, \dots W_1$, where the components of the matrices W_1 to W_L are drawn i.i.d from a Gaussian. Characterizing such density function is, in general intractable, or otherwise very difficult. However, in the asymptotic case where $dim(W) \to \infty$ and W is square, the density function admits the following concise closed-form (Eq. 13 of Pennington et al. (2017) derived from Neuschel (2014)):

$$p(\sigma(\phi)) = \frac{2}{\pi}\sqrt{\frac{\sin^3(\phi)\sin^{L-2}(L\phi)}{\sin^{L-1}((L+1)\phi)}} \quad \sigma(\phi) = \sqrt{\frac{\sin^{L+1}((L+1)\phi)}{\sin(\phi)\sin^{L}(L\phi)}}\,, \tag{5}$$

where σ denotes singular values (parameterized by $\phi \in [0, \frac{\pi}{L+1}]$) and p denotes the probability density function of σ for $\sigma \in [0, \sigma_{\max}]$, and $\sigma^2_{\max} = L^{-L}(L+1)^{L+1}$. The parametric probability density function spans the whole singular value spectrum when sweeping the variable ϕ.

We are interested in computing the effective rank of W. Using the above density function, we can write it in the form:

$$\rho(W) = -\int_0^{\sigma_{\max}} \frac{\sigma}{c} \log(\frac{\sigma}{c}) p(\sigma)\, d\sigma\,, \tag{6}$$

We now write this integral in terms of ϕ as the integration variable, such that we can leverage the density function in Eqn. 5. Using the change of variable, we have:

$$\rho(W;L) \quad = \quad -\int_0^{\frac{\pi}{L+1}} \frac{\sigma(\phi)}{c} \log(\frac{\sigma(\phi)}{c})\big(-p(\sigma(\phi))\sigma'(\phi)\big)\, d\phi\,, \tag{7}$$

where $\sigma'(\phi) = \frac{d}{d\phi}\sigma(\phi)$. Note that the integral limits $[0, \sigma_{\max}]$ on σ respectively translate[4] into $[\frac{\pi}{L+1}, 0]$ on ϕ. Computing the inner probability term, we get:

$$-p(\sigma(\phi))\sigma'(\phi) = \frac{1}{2\pi}\Big(1 + L + L^2 - L(L+1)\cos(2\phi) - (L+1)\cos(2L\phi) + L\cos(2(1+L)\phi)\Big)\csc^2(L\phi)\,.$$

As $\rho(W;L)$ is differentiable in L, $\rho(W;L)$ decreases in L if and only if $\frac{d\rho}{dL} < 0$. Since integration and differentiation are w.r.t. different variables, they commute; we can first compute the derivative of the integrand w.r.t. L and then integrate w.r.t. ϕ and show that the result is negative.

The integrand can be expressed as:

$$-\frac{\sigma(\phi)}{c}\log(\frac{\sigma(\phi)}{c})\big(-p(\sigma(\phi))\sigma'(\phi)\big) \;=\; hv^2\log(u)\sqrt{u}\,, \tag{8}$$

[4]note that the direction of integration needs to flip (by multiplying by -1) to account for flip of the upper and lower limits.

where the variables h, v and u are defined as:

$$
\begin{aligned}
h &= \frac{1}{4\pi}\Big(1 + L(L+1)(1-\cos(2\phi)) - (1+L)\cos(2L\phi) + L\cos(2(1+L)\phi)\Big) \\
v &= \csc(L\phi) \qquad (9) \\
u &= \frac{L^L}{(L+1)^{L+1}} \csc(\phi)\sin^{-L}(L\phi)\sin^{L+1}((L+1)\phi)\,. \qquad (10)
\end{aligned}
$$

Now it is straightforward to differentiate the integrand and show that it has the following form:

$$
\frac{d}{dL} hv^2 \log(u)\sqrt{u} = \frac{v(h(2+\log(u))vu' + 2\log(u)u(vh' + 2hv'))}{2\sqrt{u}} \qquad (11)
$$

We should now integrate the above form w.r.t. ϕ from 0 to $\frac{\pi}{L+1}$ and examine the sign of the integrated form. The Eqn. 11 is positive for smaller ϕ and negative for larger ϕ. Furthermore, the area under the positive region is smaller than the negative one; hence, the total value of the integral is negative. This completes the proof that the singular values decrease as a function of the number of layers.

The proof here considers the asymptotic case when $\dim(W) \to \infty$. This limit case allowed us to use the probability distribution of the singular values. Although we do not provide proof for the finite case, our results demonstrate that it holds empirically in practice (see Figure 1).

H LEAST-SQUARES LEARNING DYNAMICS

The learning dynamics of a linear network change when over-parameterized. Here, we derive the effective update rule on least-squares using linear neural networks to provide motivation on why they have differing update dynamics. For a single-layer linear network parameterized by W, without bias, the update rule is:

$$W^{(t+1)} \leftarrow W^{(t)} - \eta \nabla_{W^{(t)}} \mathcal{L}(W^{(t)}, x, y) \tag{12}$$
$$= W^{(t)} - \eta \nabla_{W^{(t)}} \frac{1}{2}(y - W^{(t)}x)^2 \tag{13}$$
$$= W^{(t)} - \eta(W^{(t)}xx^T - yx^T) \tag{14}$$

Where η is the learning rate. Similarly, the update rule for the two-layer network $y = W_e x = W_2 W_1 x$ can be written as:

$$W_1^{(t+1)} \leftarrow W_1^{(t)} - \eta(W_2^{(t)})^T(W_e^{(t)}xx^T - yx^T) \tag{15}$$
$$W_2^{(t+1)} \leftarrow W_2^{(t)} - \eta(W_e^{(t)}xx^T - yx^T)(W_1^{(t)})^T \tag{16}$$
$$\tag{17}$$

Using a short hand notation for $\nabla \mathcal{L}^{(t)} = W_e^{(t)}xx^T - yx^T$, we can compute the effective update rule for the two-layer network:

$$W_e^{(t+1)} = W_2^{(t+1)} W_1^{(t+1)} \tag{18}$$
$$= W_e^{(t)} - \overbrace{\eta(W_2^{(t)} W_2^{(t)T} \nabla \mathcal{L}^{(t)} + \nabla \mathcal{L}^{(t)} W_1^{(t)T} W_1^{(t)})}^{\text{first order } \mathcal{O}(\eta)} + \overbrace{\eta^2 \nabla \mathcal{L}^{(t)} W_e^{(t)T} \nabla \mathcal{L}^{(t)}}^{\text{second order } \mathcal{O}(\eta^2)} \tag{19}$$
$$\approx W_e^{(t)} - \eta(P_2 \nabla \mathcal{L}^{(t)} + \nabla \mathcal{L}^{(t)} P_1^T) \tag{20}$$

Where $P_i^{(t)} = W_i^{(t)} W_i^{(t)T}$ are the preconditioning matrices. The higher order terms can be ignored if the step-size is chosen sufficiently small.

(General case) For a linear network with d-layer expansion, the update for layer $1 \leq i \leq d$ is:

$$W_i^{(t+1)} \leftarrow W_i^{(t)} - \eta \overbrace{(W_d^{(t)} \cdots W_{i+1}^{(t)})^T}^{\text{weights} > i} \overbrace{(W_e^{(t)}xx^T - yx^T)}^{\text{original gradient}} \overbrace{(W_{i-1}^{(t)} \cdots W_1^{(t)})^T}^{\text{weights} < i} \tag{21}$$

Denoting $W_{j:i} = W_j \cdots W_{i+1} W_i$ for $j > i$, the effective update rule for the end-to-end matrix is:

$$W_e^{(t+1)} = \prod_{1<i<d} W_i^{(t+1)} = \prod_{1<i<d} (W_i - \eta W_{d:i+1}^{(t)T} \nabla \mathcal{L}^{(t)} W_{i-1:1}^T) \tag{22}$$
$$= W_e^{(t)} - \eta \sum_{1<i<d} W_{d:i+1} W_{d:i+1}^T \nabla \mathcal{L}^{(t)} W_{i-1:1}^T W_{i-1:1} + \mathcal{O}(\eta^2) + \cdots + \mathcal{O}(\eta^d) \tag{23}$$
$$\approx W_e^{(t)} - \eta \sum_{1<i<d} \underbrace{W_{d:i+1} W_{d:i+1}^T}_{\text{left precondition}} \underbrace{\nabla \mathcal{L}^{(t)}}_{\text{original gradient}} \underbrace{W_{i-1:1}^T W_{i-1:1}}_{\text{right precondition}} \tag{24}$$

The update rule for the general case has a much more complicated interaction of variables. For the edge $i = 1$ and $i = p$ the left and right preconditioning matrix is an identity matrix respectively.

I RANK-LANDSCAPE

We visualize the effective rank landscape of the effective weights in Figure 10 and Gram matrices in Figure 11. We use single and two-layer linear networks for effective-rank landscape. We use two-layer, and four-layer ReLU networks for the Gram matrix and are constructed from 128 randomly sampled input data. For both methods, all the weights are sampled from the same distribution. The landscape is constructed by moving along random directions u, v. We observe that over-parameterized linear and non-linear models almost always exhibit a lower-rank landscape than their shallower counterparts.

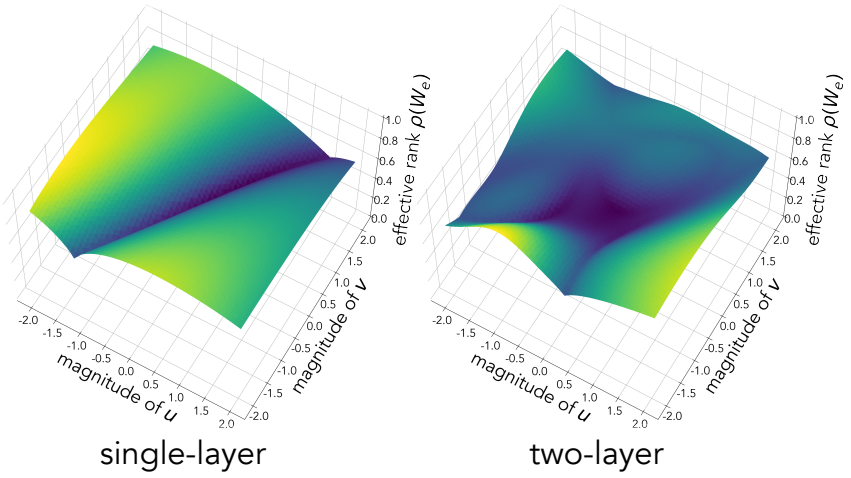


Figure 10: **Rank landscape**: The landscape of the effective rank ρ of a linear function W_e parameterized either by a single-layer network ($W_e = W$) or a two-layer linear network ($W_e = W_2 W_1$). The visualization illustrates a simplicity bias of depth, where the two-layer model has relatively more parameter volume mapping to lower rank W_e. Both models are initialized to the same end-to-end weights W_e at the origin. Motivated by Goodfellow et al. (2015), the landscapes are generated using 2 random parameter directions u, v to compute $f(\alpha, \beta) = \rho(W + \alpha \cdot u + \beta \cdot v)$ for the single-layer model and $f(\alpha, \beta) = \rho((W_2 + \alpha \cdot u_2 + \beta \cdot v_2) \cdot (W_1 + \alpha \cdot u_1 + \beta \cdot v_1))$ for the two-layer model ($u = [u_1, u_2]$, $v = [v_1, v_2]$).

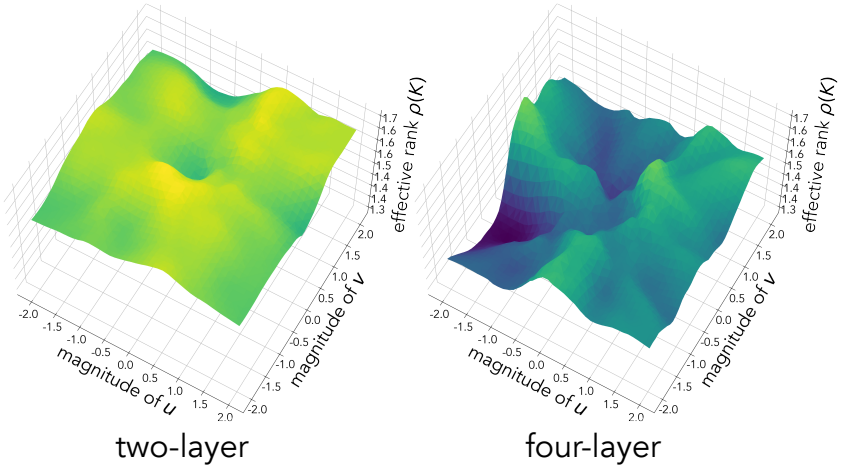


Figure 11: **Kernel rank landscape**: The landscape of the effective rank ρ computed on the kernels constructed from random features.

J GRAM MATRIX AND NON-LINEARITIES

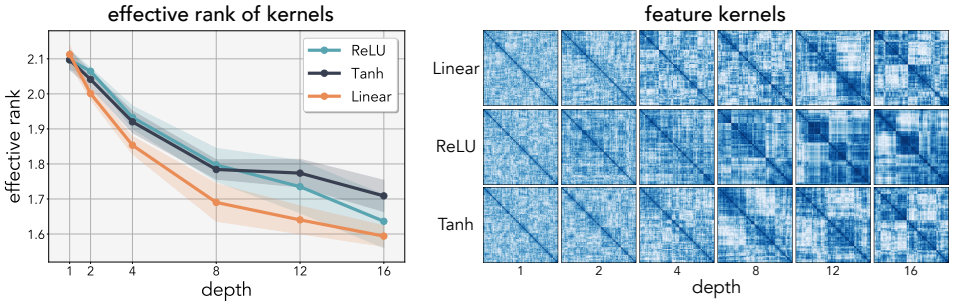


Figure 12: **Gram matrices of networks:** The Gram matrix for linear and non-linear model with rectified (ReLU) and logistic (Tanh) non-linearities.

In Figure 12, we visualize the effective rank of the Gram matrix using the same least-squares setup as we vary the depth of the model. The Gram matrix is computed using the cosine-distance on the features of the test-set after achieving zero-training loss. The effective rank is computed on the Gram matrix for both linear and non-linear models with rectified (ReLU) and logistic (Tanh) non-linearities. Increasing the number of layers decreases the effective rank of the Gram matrix. We observed the emergence of block structures in the Gram matrix as we increased the number of layers, indicating the embeddings become more low-rank with depth. This observation is also visually indicated on Figure 12 (right) with the emergence of the block-structures in the hierarchically clustered Gram matrices Rokach & Maimon (2005).

K RELATIONSHIP BETWEEN WEIGHT AND EMBEDDINGS

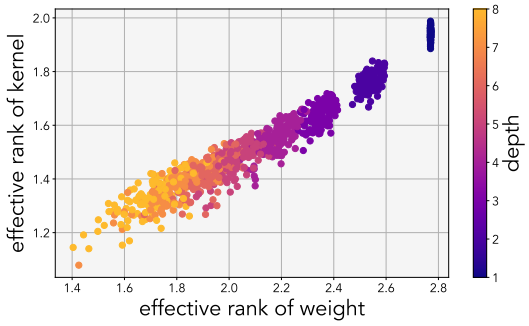


Figure 13: **Rank relation of kernel and weight**: Each point represents randomly drawn network. For each network, we compute the rank on the effective weight and also the linear kernel. The kernel is constructed from the MNIST dataset. The rank of the kernels and weights have a linear relationship.

We show that there is an almost one-to-one relationship between the effective rank of the weights and the effective rank of the Gram matrices in deep linear models. The figure plots this relationship for random deep linear networks applied to random subsets of the MNIST dataset. Moreover, it becomes apparent that the number of layers dictates the rank of the embedding as well as the weights.