CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 134 **March 28, 2022**

SGD Noise and Implicit Low-Rank Bias in Deep Neural Networks

Tomer Galanti and Tomaso Poggio
Center for Brains, Minds, and Machines, MIT, Cambridge, MA, USA

Abstract

We analyze deep ReLU neural networks trained with mini-batch stochastic gradient decent and weight decay. We prove that the source of the SGD noise is an implicit low rank constraint across all of the weight matrices within the network. Furthermore, we show, both theoretically and empirically, that when training a neural network using Stochastic Gradient Descent (SGD) with a small batch size, the resulting weight matrices are expected to be of small rank. Our analysis relies on a minimal set of assumptions and the neural networks may include convolutional layers, residual connections, as well as batch normalization layers.

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

SGD Noise and Implicit Low-Rank Bias in Deep Neural Networks

Tomer Galanti and Tomaso Poggio
Center for Brains, Minds and Machines, MIT
Massachusetts Institute of Technology
MA, USA

Abstract

We analyze deep ReLU neural networks trained with mini-batch Stochastic Gradient Descent (SGD) and weight decay. We study the source of SGD noise and prove that when training with weight decay, the only convergence points of SGD are zero functions. Furthermore, we show, both theoretically and empirically, that when training a neural network using SGD with a small batch size, the resulting weight matrices are expected to be of small rank. Our analysis relies on a minimal set of assumptions and the neural networks may include residual connections, as well as batch normalization layers.

1 Introduction

Over the past few years, neural networks have challenged machine learning theory with several mysteries. We consider two of them here that are, to our knowledge, so far unsolved. The first is about the mathematical reasons for a bias in Stochastic Gradient Descent (SGD) towards neural networks with low-rank weight matrices, a fact which has been empirically observed (see Timor et al. [2022] and references therein). The second is regarding the role (see Li et al. [2021] and references therein) and origin of SGD noise. The origin of the SGD noise represents an especially important and rather neglected question. In this paper, we report a simple observation that solves both mysteries and, somewhat surprisingly, shows that they have the same mathematical root. The observation is that the matrix $\frac{\partial f_W}{\partial W^i}$ is a matrix of rank ≤ 1, where f_W is a deep ReLU network with a scalar output and weight matrices $\{W^i\}_{i\in I}$ (I is some finite set of indices)[1].

1.1 Related Work

Stochastic gradient descent (SGD) is widely used for optimizing deep models and has become a standard approach in the field [Bottou, 1991]. Originally designed to address the computational challenges of gradient descent (GD), recent research indicates that SGD also introduces an important implicit regularization effect. This regularization prevents overparameterized models from converging to suboptimal minima that may not generalize well [Zhang et al., 2016, Jastrzębski et al., 2017, Keskar et al., 2017, Zhu et al., 2019].

Empirical studies have revealed several key findings about SGD. First, SGD outperforms GD in various scenarios Zhu et al. [2019]. Second, small-batch SGD tends to generalize better than large-batch SGD [Hoffer et al., 2017, Keskar et al., 2017]. Third, gradient descent with additional noise fails to match the performance of SGD Zhu et al. [2019]. However, despite extensive research efforts, the implicit regularization induced by the noise in SGD remains not fully understood. Furthermore,

[1]When f_W outputs a k-dimensional vector, each output component of the tensor corresponding to $\frac{\partial f}{\partial W^i}$ is a matrix of rank at most 1.

Preprint. Under review.

the source of this noise during SGD training, which persists even for extended periods long after the presumed convergence of the process, remains a topic that has received little attention and lacks comprehensive understanding. A separate focus of considerable research in recent years has been the implicit bias in linear neural networks towards rank minimization. Most of the interest was on the matrix factorization problem, which corresponds to training a depth-2 linear neural network with multiple outputs w.r.t. the square loss. As described by Timor et al. [2022], Gunasekar et al. [2017] initially conjectured that the implicit regularization in matrix factorization can be characterized by the nuclear norm of the corresponding linear predictor. This conjecture was formally refuted by Li et al. [2020]. Razin and Cohen [2020] conjectured that the implicit regularization in matrix factorization can be explained by rank minimization, and also hypothesized that some notion of rank minimization may be key to explaining generalization in deep learning. Li et al. [2020] established evidence that the implicit regularization in matrix factorization is a heuristic for rank minimization. Beyond factorization problems, Ji and Telgarsky [2020] showed that in linear networks of output dimension 1, gradient flow (GF) w.r.t. exponentially-tailed classification losses converges to networks where the weight matrix of every layer is of rank 1.

However, with nonlinear neural networks, things are less clear. Empirically, a series of papers [Denton et al., 2014, Alvarez and Salzmann, 2017, Tukan et al., 2021, Yu et al., 2017, Arora et al., 2018] showed that replacing the weight matrices by low-rank approximations results in only a small drop in accuracy. This suggests that the weight matrices in practice may be close to low-rank matrices. However, whether they provably behave this way remains unclear. Timor et al. [2022] show that for nonlinear ReLU networks, GF does not minimize rank. They also argue that ReLU networks of sufficient depth can have low-rank solutions under ℓ_2 norm minimization. This interesting result, however, applies to layers added to a network that already solves the problem and may not have any low-rank bias. It is not directly related to the mechanism described in this paper, which is more generic and applies to all layers in the network.

1.2 Contributions

We provide a mathematical analysis of the origin of SGD noise and of the implicit rank-minimization of SGD. We analyze deep ReLU networks trained with mini-batch stochastic gradient descent and regularization (i.e., weight decay). Our analysis is fairly generic: the neural networks may include linear layers, residual connections, and batch normalization layers. In addition, the loss function is only assumed to be differentiable. The central contributions are:

- In Props. 1-2, we study the source of SGD noise. The result shows that SGD noise must be always present, even asymptotically, regardless of the batch size, as long as we incorporate weight decay. Thus, SGD noise is a generic property; this also means that, generically, there is never convergence of SGD.
- In Thm. 1 we prove that mini-batch SGD has an implicit bias towards networks with low-rank matrices. This theorem connects batch size, weight decay, optimization and rank.
- In Sec. 3.2, we theoretically predict that the batch size, weight decay and learning rate act as low-rank regularizers. These predictions are later validated empirically in Sec. 4.

2 Problem Setup

We consider the problem of training a model for a standard classification or regression problem. Formally, the target task is defined by a distribution P over samples $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^n$ is the instance space, and $\mathcal{Y} \subset \mathbb{R}^k$ is a label space.

A function $f_W \in \mathcal{F} \subset \{f' : \mathcal{X} \to \mathbb{R}^k\}$ assigns a prediction to an input point $x \in \mathcal{X}$, and its performance on the distribution P is measured by the risk

$$L_P(f_W) = \mathbb{E}_{(x,y)\sim P}[\ell(f_W(x), y)], \tag{1}$$

where $\ell : \mathbb{R}^k \times \mathcal{Y} \to [0, \infty)$ is a non-negative, differentiable, loss function (e.g., MSE or cross-entropy losses). For simplicity, throughout the paper, we focus on the case where $k = 1$.

Since we do not have direct access to the full population distribution P, the goal is to learn a predictor, f_W, from some balanced training data $S = \{(x_i, y_i)\}_{i=1}^m$ of independent and identically distributed

(i.i.d.) samples drawn from P along with regularization that controls the complexity of the learned model. Namely, we intend to minimize the regularized empirical risk

$$L_S^\lambda(f_W) = \frac{1}{m}\sum_{i=1}^{m} \ell(f_W(x_i), y_i) + \lambda\|W\|_2^2, \tag{2}$$

where $\lambda > 0$ is predefined hyperparameter, controlling the amount of regularization.

2.1 Architectures and Training

In this work, the function f_W represents a neural network, consisting of a set of layers of weights interlaced with ReLU units. We employ a fairly generic definition of a neural network, that includes linear layers, residual connections and batch normalization layers. To include normalization layers, we extend f_W's input to be $(x; \tilde{\mathcal{X}})$, where $x \in \mathbb{R}^n$ and $\tilde{\mathcal{X}} = \{x_{ij}\}_{j=1}^B \subset \mathbb{R}^n$ is a batch of samples (see details below). For simplicity, $f_W(x; \tilde{\mathcal{X}})$ may denote a network with or without batch normalization layers, and specifically, $f_W(x)$ denotes a network without normalization layers. When $\tilde{\mathcal{X}}$ is clear from context, we write $\tilde{f}_W(x) := f_W(x; \tilde{\mathcal{X}})$.

Optimization. In this work, we consider optimizing f_W using mini-batch stochastic gradient descent (SGD), possibly with detaching certain layers. Namely, we aim at minimizing the regularized empirical risk $L_S(f_W)$ by applying SGD for a certain number of iterations. We initialize W_0 using a standard initialization procedure and iteratively update W_t. At each iteration, we sample a subset $\tilde{S} = \{(x_{ij}, y_{ij})\}_{j=1}^B \subset S$ uniformly at random and update $W_{t+1} \leftarrow W_t - \eta_t \cdot \frac{\bar{\partial} L_{\tilde{S}}(f_{W_t}(\,\cdot\,; \tilde{\mathcal{X}}))}{\partial W}$, where $\tilde{\mathcal{X}} = \{x_{ij}\}_{j=1}^B$ and $\eta_t > 0$ is our learning rate at iteration t. Here, $f_W(\,\cdot\,; \tilde{\mathcal{X}}) : \mathbb{R}^n \to \mathbb{R}$ is the function given by $f_W(x; \tilde{\mathcal{X}})$ for a fixed value of $\tilde{\mathcal{X}}$. Furthermore, for a given function $g_\theta = g_\theta^L \circ \ldots g_\theta^1 : \mathbb{R}^n \to \mathbb{R}^k$ where $\{g_\theta^{i_j}\}_{j=1}^r$ are detached, $\frac{\bar{\partial} g_\theta(z)}{\partial \theta}$ denotes the pseudo gradient of g_θ, which is computed using the chain rule, while treating $\{g_\theta^{i_j}\}_{j=1}^r$ as independent of θ.

Graph structure. Formally, f_W is as a directed acyclic graph (DAG) $G = (V, E)$, where $V = \{v_1, \ldots, v_L\}$ consists of the layers within the network and each edge $e_{ij} = (v_i, v_j) \in E$ specifies a connection between two layers. Each layer is a function $v_i : \mathbb{R}^{n \to d_i}$ and each connection holds a weight $W^{ij} \in \mathbb{R}^{d_i \times d_j}$. The layers are divided into three categories: the input layer v_1, the output layer and intermediate layers. The output layer is not connected to any other layer and no one of the layers connects with the input layer. Each layer v_i is evaluated as follows

$$v_i(x; \tilde{\mathcal{X}}) = \sigma(n_i(\sum_{j \in \text{pred}(i)} W^{ij} \cdot v_j(x; \tilde{\mathcal{X}}); \tilde{\mathcal{X}})), \tag{3}$$

except for the last layer v_L that computes

$$f_W(x; \tilde{\mathcal{X}}) = v_L(x; \tilde{\mathcal{X}}) = \sum_{j \in \text{pred}(L)} W^{Lj} \cdot v_j(x; \tilde{\mathcal{X}}), \tag{4}$$

where $\text{pred}(i)$ the set of indices j, such that $(v_i, v_j) \in E$. Here, σ is an element-wise activation function (e.g., ReLU, Leaky ReLU, sigmoid) and n_i is either the identity function $n_i(z; \tilde{\mathcal{X}}) = z$ or batch normalization (see below).

In this work, the weight matrices W^{pq} could be trainable, $(v_p, v_q) \in E_T$ (e.g., fully-connected layers), or constant, $(v_p, v_q) \notin E_T$ (e.g., residual connections). Throughout the analysis, we consider paths within the graph G, denoted by $\pi = (\pi_0, \ldots, \pi_T)$, where $\pi_0 = 1$, $\pi_T = L$ and for all $i = 0, \ldots, T-1 :\ (v_{\pi_i}, v_{\pi_{i+1}}) \in E$.

Normalization layers. The neural networks may also include batch normalization layers Ioffe and Szegedy [2015]. Namely,

$$\begin{aligned} n_l(v_l(x; \tilde{\mathcal{X}}); \tilde{\mathcal{X}}) &= \gamma_l \cdot A_l(\tilde{\mathcal{X}}) \cdot (v_l(x; \tilde{\mathcal{X}}) - b_l(\tilde{\mathcal{X}})) + \mathbb{1} \cdot \beta_l \\ A_l(\tilde{\mathcal{X}}) &= \text{diag}\left(\left(\sqrt{\text{Var}_{\hat{x} \sim U[\tilde{\mathcal{X}}]}(v_l(\hat{x}; \tilde{\mathcal{X}})_j)}\right)^{-1}\right)_{j=1}^{d_l} \\ b_l(\tilde{\mathcal{X}}) &= \mathbb{E}_{\hat{x} \sim U[\tilde{\mathcal{X}}]}[v_l(\hat{x}; \tilde{\mathcal{X}})], \end{aligned} \tag{5}$$

where $\gamma_l, \beta_l \in \mathbb{R}$ are trainable scale and shift parameters and $\mathbb{1} := (1, \ldots, 1)$. To simplify the analysis, we detach (also known as stop-grad [Grill et al., 2020]) the matrices $A_l(\tilde{\mathcal{X}})$ during backpropagation at any step during training. Namely, we treat $\frac{\bar{\partial} A_l(\tilde{\mathcal{X}})}{\partial W} = 0$ in order to compute the training updates. We note that detaching is a well-established approach for training a network with normalization [Wiesler et al., 2014, Ioffe, 2017, Raiko et al., 2012, Povey et al., 2014, Xu et al., 2019, Gouk et al., 2021]. In fact, Xu et al. [2019] showed that detaching the variance term of layer normalization is competitive and even improves the performance on various tasks.

At test time, to evaluate a given network on a given input x, we use a $f(x; \hat{\mathcal{X}})$, where $\hat{\mathcal{X}} = \{x_i\}_{i=1}^m$ is the full training data. We note that the evaluation process of the network with this kind of normalization is the same as with standard batch normalization layers.

3 Theoretical Results

In this section, we describe the main theoretical results of this work. In Section 3.1, we show that the rank one constraint is the source of SGD noise. In Section 3.2 we provide a brief analysis showing that mini-batch SGD implicitly learns neural networks with low-rank matrices. Proofs are provided in Appendix B.

3.1 Degeneracy and the Origin of SGD Noise

The term 'SGD noise' refers to the inherent inability of SGD to converge to a stationary solution for the weight matrices that is the same across different mini-batches. In this section, we characterize the convergence of mini-batch SGD. Our results are essentially impossibility results that show that convergence of SGD takes place only when the network is zero. This implies that asymptotic noise is generically unavoidable in practical cases when training is successful. For simplicity, we focus on normalization-free univariate neural networks $f_W : \mathbb{R}^n \to \mathbb{R}$ and assume that $x_i \neq 0$ for all $i \in [m]$.

We start by spelling out the conditions for convergence of SGD. We note that at convergence, we have $\nabla_{W^{pq}} L_{\tilde{S}}(f_W) = 0$ for all pairs $(v_p, v_q) \in E_T$ and mini-batches $\tilde{S} \subset S$ of size $B < m$. Specifically, we we can write

$$0 \;=\; \nabla_{W^{pq}} L^{\lambda}_{\tilde{S}}(f_W) \;=\; \frac{1}{B} \sum_{(x,y)\in\tilde{S}} \frac{\partial \ell(f_W(x), y)}{\partial f_W(x)} \cdot \frac{\partial f_W(x)}{\partial W^{pq}} + 2\lambda W^{pq}. \tag{6}$$

Suppose we have two batches $\tilde{S}_1, \tilde{S}_2 \subset S$ of size B that differ by only one sample. We denote the unique sample of each batch by (x_i, y_i) and (x_j, y_j) respectively. We notice that

$$\begin{aligned} 0 \;&=\; \nabla_{W^{pq}} L^{\lambda}_{\tilde{S}_1}(f_W) - \nabla_{W^{pq}} L^{\lambda}_{\tilde{S}_2}(f_W) \\ &=\; \frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \cdot \frac{\partial f_W(x_i)}{\partial W^{pq}} - \frac{\partial \ell(f_W(x_j), y_j)}{\partial f_W(x_j)} \cdot \frac{\partial f_W(x_j)}{\partial W^{pq}}. \end{aligned} \tag{7}$$

Therefore, we conclude that for all $i, j \in [m]$

$$M^{pq} \;=\; \frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \cdot \frac{\partial f_W(x_i)}{\partial W^{pq}} \;=\; \frac{\partial \ell(f_W(x_j), y_j)}{\partial f_W(x_j)} \cdot \frac{\partial f_W(x_j)}{\partial W^{pq}}. \tag{8}$$

Hence, for all $(v_p, v_q) \in E_T$ and $i \in [m]$,

$$\frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \cdot \frac{\partial f_W(x_i)}{\partial W^{pq}} + 2\lambda W^{pq} \;=\; M^{pq} + 2\lambda W^{pq} \;=\; 0. \tag{9}$$

Therefore, unless $\lambda = 0$ or $\forall\ (v_p, v_q) \in E_T :\ W^{pq} = 0$, we conclude that $\frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \neq 0$ for all $i \in [m]$. In this case, we also obtain that $\{\frac{\partial f_W(x_i)}{\partial \mathrm{vec}(W^{pq})}\}_{i=1}^m$ are collinear vectors by Eq. 8.

Therefore, any convergence point of training a neural network using mini-batch SGD along with weight decay is highly degenerate and does not perfectly fit any one of the training labels. To better understand the essence of this degeneracy, we provide the following proposition, which is specialized for ReLU networks.

Proposition 1 ($\lambda > 0$)**.** *Let $\ell(a,b)$ be a differentiable loss function, $\lambda > 0$, and let $f_W(x)$ be a ReLU neural network, where $\{(v_p, v_1) \in E \mid v_p \in V\} \subset E_T$. Let W be a convergence point of mini-batch SGD for minimizing $L_S^\lambda(f_W)$. Then, either $f_W \equiv 0$ or $\{x_i\}_{i=1}^m$ are collinear vectors.*

The above proposition shows that unless the training samples lie on a one-dimensional linear space, at convergence, SGD learns the zero function when training with weight decay. Since both cases are unrealistic, we conclude that convergence is impossible in any practical scenario.

In the following proposition, we look at the convergence of SGD when training without weight decay.

Proposition 2 ($\lambda = 0$)**.** *Let $\ell(a,b)$ be a differentiable loss function, $\lambda = 0$, and let $f_W(x)$ be a ReLU neural network, where $\{(v_p, v_1) \in E \mid v_p \in V\} \subset E_T$. Let W be a convergence point of mini-batch SGD for minimizing $L_S^\lambda(f_W)$. Assume that $\{x_i\}_{i=1}^m$ are not collinear. Then, for any $i \in [m]$, for which $\frac{\partial \ell(f_W(x_i),y_i)}{\partial f_W(x_i)} \neq 0$, we have, $f_W(x_i) = 0$. In particular, if ℓ is convex and $\forall\, b \in \mathbb{R} \nexists\, a^* \in \mathbb{R} :\ \ell(a^*, b) = \inf_a \ell(a,b)$, then, $\forall\, i \in [m] :\ f_W(x_i) = 0$.*

The above proposition shows that at convergence, the neural network outputs zero for any training sample that it does not perfectly fit (i.e., $\frac{\partial \ell(f_W(x_i),y_i)}{\partial f_W(x_i)} \neq 0$). In particular, if ℓ is an exponential-type loss function (e.g., binary cross-entropy, logistic loss and exponential loss), $\ell(a,b)$ has no minima a for any $b \in \mathbb{R}$, and therefore, the only possible convergent points are ones for which $f_W(x_i) = 0$ across all $i \in [m]$. While theoretically convergence to a non-zero function is not guaranteed, in practice training without weight decay can still fall into the regime of 'almost convergence', in which $\max_{i\in[m]} \left|\frac{\partial \ell(f_W(x_i),y_i)}{\partial f_W(x_i)}\right|$ is tiny and the training steps $-\eta \cdot \frac{\partial \ell(f_W(x_i),y_i)}{\partial f_W(x_i)} \cdot \frac{\partial f_W(x_i)}{\partial W^{pq}}$ are very small as a result. In general, by increasing the size of the network (i.e., the width or the depth), we should expect the term $\max_{i\in[m]} \left|\frac{\partial \ell(f_W(x_i),y_i)}{\partial f_W(x_i)}\right|$ to tend to zero (if the network is trained successfully). On the other hand, with the MSE loss we may have convergence when $\forall\, i \in [m] :\ f_W(x_i) = y_i$ (in this case $\forall\, i \in [m] :\ \frac{\partial \ell(f_W(x_i),y_i)}{\partial f_W(x_i)} = 0$).

It is worth noting that our analysis critically depends on using SGD instead of GD during optimization. While many papers analyze the training dynamics and critical points of GD, the solutions of SGD are highly degenerate and will, in general, look different from the solutions of GD. For instance, it is impossible to derive Eq. 8 without relying on optimization through mini-batch SGD, and without weight decay, we could not argue that W^{pq} is proportional to the low-rank matrix M^{pq}. Finally, we note that the analysis above is independent of the batch size as long as it is strictly smaller than the full dataset's size. As follows from Eq. 9 the convergence points of mini-batch SGD are equivalent to the convergence points of SGD with batch size 1.

3.2 Implicit Rank Minimization

In the previous section we showed that at convergence, the neural network is highly degenerate and the trainable matrices along the network are zero matrices. As a next step, we extend the discussion and study rank minimization in non-convergent points. Specifically, we demonstrate an implicit bias of mini-batch SGD to learn weight matrices of small rank, depending on the batch size, the weight decay and its ability to successfully minimize the objective function. In the following theorem, we introduce an upper bound on the distance of the various weights matrices within the network from matrices of rank $\leq B^*$ as a function of the progress of the optimization. The theorem holds for neural network with or without batch normalization.

Theorem 1. *Let $\|\cdot\|$ be any matrix norm. Let $f_W(x, \tilde{\mathcal{X}})$ be a ReLU neural network and W^{pq} be a given matrix within f_W and let $B^* < m$ be a natural number. Then,*

$$\min_{\substack{V \in R^{d_p \times d_q}: \\ \text{rank}(V) \leq B^*}} \|W^{pq} - V\| \ \leq\ \frac{1}{2\lambda} \min_{\tilde{S} \subset S:\ |\tilde{S}| = B^*} \|\nabla_{W^{pq}} L_{\tilde{S}}^\lambda(\tilde{f}_W)\| \tag{10}$$

The theorem above provides an upper bound on the minimal distance between the various weight matrices in the network W^{pq} and matrices of rank at most B^*. The upper bound is proportional to the minimal norm (w.r.t. the selection of the batch) of the gradient of a regularized risk evaluated on a batch of size B^* and essentially depends on the training hyperparameters (e.g., batch size, learning rate, weight decay, etc'). Therefore, it can be used to draw predictions about the relationships between certain hyperparameters, the rank and the performance.

Several interesting predictions follow the theorem above.

Rank and batch size. We note that SGD training with batches of size B directly minimizes the following objective $\mathrm{Avg}_{\tilde{S}\subset S:\,|\tilde{S}|=B}[\|\nabla_{W^{pq}} L^{\lambda}_{\tilde{S}}(f_W)\|]$ (for all $(p,q)\in E$). This quantity upper bounds the term $\min_{\tilde{S}\subset S:\,|\tilde{S}|=B}\|\nabla_{W^{pq}} L^{\lambda}_{\tilde{S}}(f_W)\|$ from Equation 10. Therefore, if training successfully minimizes $\mathrm{Avg}_{\tilde{S}\subset S:\,|\tilde{S}|=B}[\|\nabla_{W^{pq}} L^{\lambda}_{\tilde{S}}(f_W)\|]$, then, we expect the matrix W^{pq} to be close to a matrix of rank $\leq B$. Hence, we predict that the batch size regularizes the rank of the various matrices in the network (when fixing the rest of the hyperparameters). Namely, *we expect training with smaller batch sizes to produce matrices of smaller ranks*. Specifically, in contrast to SGD with small batch sizes, *we expect that GD would not regularize the rank of the network's matrices* (unless the training data is very small).

Rank and weight decay. The dependence of the bound on λ is twofold: through the multiplicative term λ^{-1} and through the term $\min_{\tilde{S}\subset S:\,|\tilde{S}|=B^*}\|\nabla_{W^{pq}} L^{\lambda}_{\tilde{S}}(f_W))\|$ that indirectly depends on λ. For two degrees of weight decay $\lambda_2 > \lambda_1 > 0$, as long as SGD minimizes $\min_{\tilde{S}\subset S:\,|\tilde{S}|=B^*}\|\nabla_{W^{pq}} L^{\lambda_i}_{\tilde{S}}(f_W)\|$ to a small value (for both λ_1,λ_2), then we expect the matrices W^{pq} to be of smaller ranks when utilizing λ_2 than when utilizing λ_1. Therefore, *we predict that the ranks of the matrices should decrease as we increase* λ. Furthermore, if $\lambda = 0$, then the bound is infinite and we lose the low-rank constraint. Therefore, *we predict that without weight decay we do not have an implicit regularization towards rank minimization.*

Rank and learning rate. Typically, unless the learning rate is not too large, when training a model with larger learning rates we should expect the objective's gradient to be of a smaller norm. Therefore, for larger values of η, we should expect $\min_{\tilde{S}\subset S:\,|\tilde{S}|=B}\|\nabla_{W^{pq}} L^{\lambda}_{\tilde{S}}(f_W)\|$ to be smaller. Therefore, similar to the batch size, we predict that the learning rate also induces a regularization effect, for which, *with larger learning rates we learn matrices of smaller rank.* Furthermore, as we discussed and observe empirically, when training with small batch sizes we expect to learn matrices of very small ranks. We note that neural networks with matrices of very low ranks typically under perform on the training set, due to their limited expressivity. Therefore, *we predict that to effectively train a model with smaller batch sizes it is necessary to decrease the learning rate accordingly*, similar to previous empirical observations Krizhevsky [2014], Goyal et al. [2018].

4 Experiments

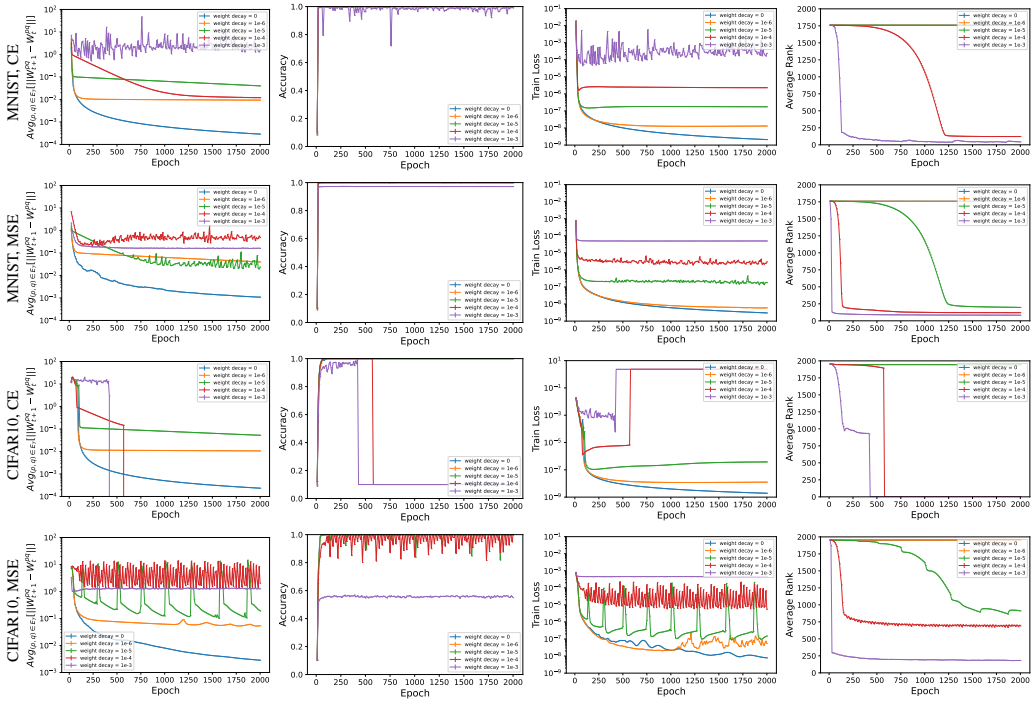
In this section we experimentally analyze the convergence (and non-convergence) of training neural networks with SGD. In addition, we empirically study the implicit bias towards low-rank in deep ReLU neural networks. Throughout the experiments we extensively vary different hyperparameters (e.g., the learning rate, weight decay, batch normalization, batch size) and study their effect on the rank of the various matrices in the network[2].

4.1 Setup

Evaluation process. Following the problem setup, we consider k-class classification problems and train a multilayered neural network $f_W = e\circ h = e\circ g^L\circ\cdots\circ g^1 : \mathbb{R}^n\to\mathbb{R}^k$ on some balanced training data S. The model is trained using cross-entropy loss minimization between its logits and the one-hot encodings of the labels. Here, $g^1,\ldots,g^L$ are the various hidden layers of the network (e.g., fully-connected layers, residual blocks), where e is its top linear layer. After each epoch we compute the rank of each one of the weight matrices in the network and the train and test accuracy rates. The estimate the rank of a given matrix, we normalize it and count how many of its singular values are $\notin [-1\mathrm{e}{-4}, 1\mathrm{e}{-4}]$. In the experiments we used the MNIST and CIFAR10 datasets.

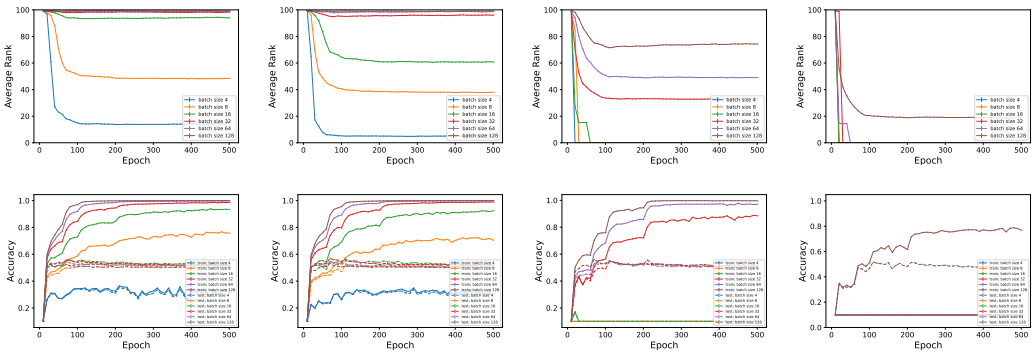
Architectures and hyperparameters. In this work we consider two types of architectures. The first architecture is a Multi-layered perceptron (MLP). Our MLPs, denoted by MLP-BN-L-H consist of L hidden layers, where each layer consists of a linear layer of width H, followed by batch normalization and ReLU. On top of that we compose a linear output layer. The second architecture is a fully-connected residual network (ResNet). Our ResNets, denoted by RES-BN-L-H consist of a linear layer of width H, followed by L residual blocks, each computing a function of the form

[2]The plots are best viewed when zooming into the pictures.



(a) (b) (c) (d)

Figure 1: **Convergence of MLP-5-2000 trained on MNIST and CIFAR10 with CE/ MSE loss.** In **(a)** we plot the averaged distance between the weight matrices at epoch t and epoch $t+1$, captured by, $\frac{1}{|E_T|}\sum_{(p,q)\in E_T}\|W_{t+1}^{pq} - W_t^{pq}\|$. In **(b)** we plot the train accuracy rates, in **(c)** we plot the averaged train loss and in **(d)** we plot the average rank across the trainable matrices.



$\eta = 0.001$ $\eta = 0.01$ $\eta = 0.1$ $\eta = 0.5$

Figure 2: **Average rank of MLP-BN-10-100 trained on CIFAR10 with varying batch sizes.** Each line stands for the results of a different batch size. In the (top) row we plot the average rank across layers during training and in the (bottom) row we plot the train and test accuracy rates for each setting.

$z + \sigma(n_2(W_2\sigma(n_1(W_1 z))))$, where $W_1, W_2 \in \mathbb{R}^{H\times H}$, n_1, n_2 are batch normalization layers and σ is the element-wise ReLU function. Again, the network is ended with a linear output layer. We denote by MLP-L-H and RES-L-H the same architectures without applying batch normalization.

Unless mentioned otherwise, we trained the models using mini-batch SGD, with momentum 0.9. The learning rate is decayed by a factor of 0.1 three times at epochs 60, 100, 200.

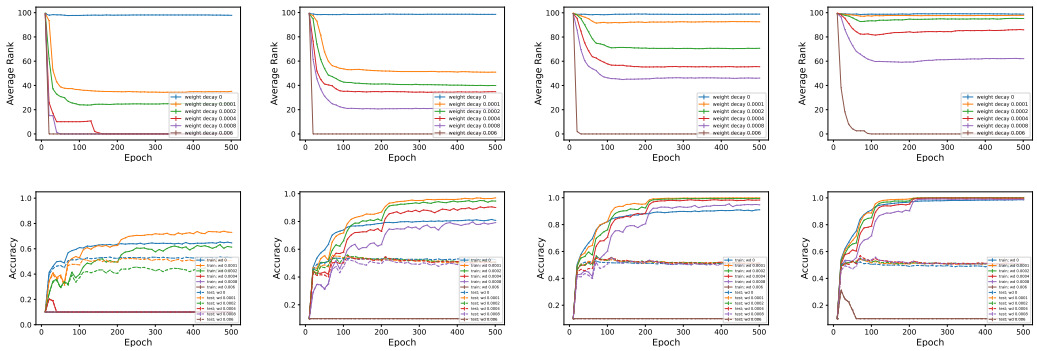


Figure 3: **Average rank of MLP-BN-10-100 trained on CIFAR10 with varying weight decay.** Each line stands for the results of a different value of λ. In the (top) row we plot the average rank across layers during training and in the (bottom) row we plot the train and test accuracy rates for each setting.

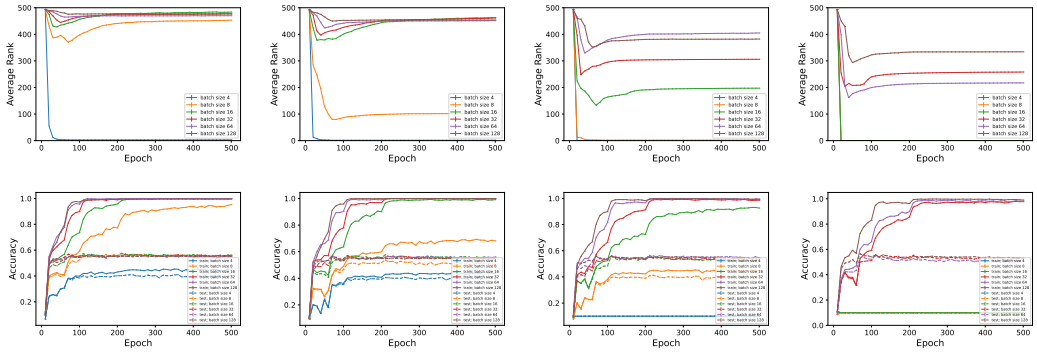


Figure 4: **Average rank of RES-BN-5-500 trained on CIFAR10 with varying batch sizes.** Each line stands for the results of a different value of batch size. In the (top) row we plot the average rank across layers during training and in the (bottom) row we plot the train and test accuracy rates for each setting.

4.2 Results on SGD Noise

In Sec. 3.1 we showed that when training a neural network with SGD, it cannot converge to a non-zero function when applying weight decay. For this purpose, we trained MLP-5-2000 networks with varying degrees of weight decay $\lambda \in \{0, 1\text{e}{-6}, 1\text{e}{-5}, 1\text{e}{-4}, 1\text{e}{-3}\}$ and investigated the degree of convergence they achieve. For simplicity, each model was trained using mini-batch SGD with batch size 128 and learning rate 0.1 for 2000 epochs, but without scheduling or momentum. We trained the models once with the CE loss and once with the squared loss.

In order to evaluate the degree of convergence of the networks, we use the following quantity:

$$d(W_{t+1}, W_t) \;:=\; \frac{1}{|E_T|} \sum_{(p,q)\in E_T} \|W_{t+1}^{pq} - W_t^{pq}\|,$$

where $\{W_t^{pq}\}_{(p,q)\in E_T}$ are the various trainable matrices in the network at epoch t. This quantity measures the averaged distance between the network's matrices at epochs t and $t+1$. Convergence is possible if and only if this quantity tends to zero.

In Fig. 1 we monitor (a) $d(W_{t+1}, W_t)$, (b) the train accuracy rates, (c) the averaged train losses and (d) the averaged rank of the trainable matrices. As predicted in Prop. 1, when training with $\lambda > 0$, W_t either converges to zero and f_{W_t} to the zero function (e.g., the averaged ranks for $\lambda = 1\text{e}{-4}, 1\text{e}{-3}$

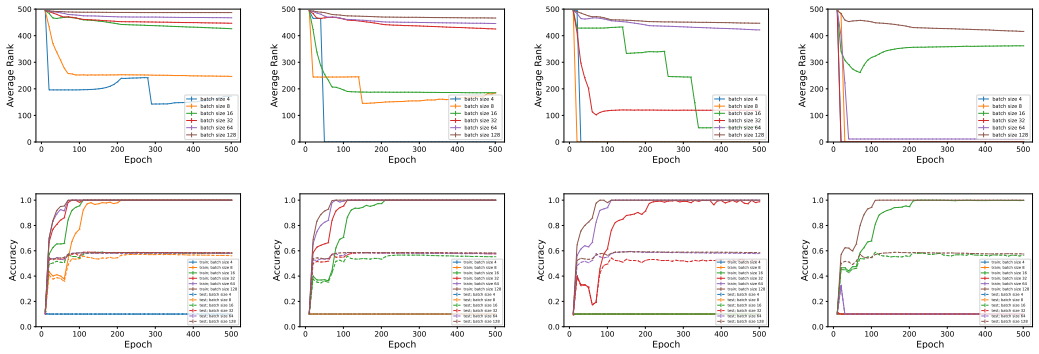


Figure 5: **Average ranks and accuracy rates of MLP-5-500 trained on CIFAR10 with varying batch sizes.** Each line stands for the results of a different value of batch size. In the (top) row we plot the average rank across layers during training and in the (bottom) row we plot the train and test accuracy rates for each setting.

when training on CIFAR10 with the CE loss are zero) or W_t is does not converge ($d(W_{t+1}, W_t)$ is clearly lower bounded by a positive constant). On the other hand, the term $d(W_{t+1}, W_t)$ (and its slope) is smaller by orders of magnitude when using $\lambda = 0$ in comparison to using $\lambda > 0$ (except for cases when the selection of $\lambda > 0$ leads to $W_t^{pq} \to 0$). Interestingly, even though in certain cases the training loss and accuracy converged, the network's parameters do not converge.

Finally, as we discussed in Sec. 3.1, by training for CE loss minimization without weight decay we may encounter the 'almost convergence' regime. In this regime, even though perfect convergence is impossible (see Prop. 2) for a fixed neural network, the term $\lim_{t\to\infty} d(W_{t+1}, W_t)$ may be as small as we wish by increasing the size of the neural network. Therefore, since the MLP-5-2000 is relatively large (compared to the dataset's size) it seems that $d(W_{t+1}, W_t)$ tends to zero.

train the network to a point where it has been practically converged and the averaged loss is tiny.

4.3 Results on Rank Minimization

Rank, batch size and the learning rate. We train instances of MLP-BN-10-100, RES-BN-5-500 and MLP-5-500 with different batch sizes and initial learning rates. For each run we trained the network for 500 epochs using mini-batch SGD with momentum 0.9, $\lambda = 5\text{e}{-4}$, batch size $B \in \{2^{i+1}\}_{i=1}^{6}$ and initial learning rate $\eta \in \{0.001, 0.01, 0.1\}$.

As can be seen in Figs. 2, 4 and 5, by increasing the batch size, we essentially strengthen the low-rank constraint over the network's matrices, which eventually leads to lower ranks. This is strongly aligned with the prediction made in Sec. 3.2 that we should expect the rank of the various matrices to be smaller when training the network with smaller batch sizes. In addition, we also notice that when increasing the learning rate, the rank minimization constraint strengthens as well, which leads to lower ranks and worse training performance. Therefore, as can be seen, to train a neural network with smaller batch sizes, one needs to decrease the learning rate as well.

Rank and λ. To investigate the effect of the weight decay on rank minimization, we trained instances of MLP-10-100 with different values of $\lambda \in \{0, 1\text{e}{-4}, 2\text{e}{-4}, 4\text{e}{-4}, 8\text{e}{-4}, 6\text{e}{-3}\}$ and batch sizes $B \in \{16, 32, 64, 128\}$. For each run we trained the network for 500 epochs using SGD with momentum 0.9 and initial learning rate 0.1.

The results are reported in Fig. 3. As can be seen, by increasing λ we typically impose stronger rank minimization constraints, as predicted in Sec. 3.2. In addition, we note that regardless of the batch size, we do not have any rank minimization when $\lambda = 0$ as predicted in Sec. 3.2.

5 Conclusions

In this work we theoretically studied the source of SGD noise and the implicit bias towards low-rank weight matrices in deep neural networks. We made the several key theoretical and empirical observations:

- We show that the batch size, weight decay and learning rate contribute to a regularizing effect on the rank of the trainable matrices of neural networks;
- Weight decay and training with mini-batches are necessary to obtain rank minimization;
- SGD noise is inevitable when applying weight decay.

However, it has not escaped our attention that these results have a rich set of implications with respect to other mysteries of deep learning. This includes questions as the reason why SGD training of networks generalizes well, the relationship between rank minimization and architectural choices (residual networks [He et al., 2017], transformers [Vaswani et al., 2017] and convolutional networks [LeCun et al., 1999]) and whether rank minimization extends beyond the standard learning settings.

Acknowledgments The rank one constraint was first mentioned in a recent version of CBMM memo 112. This paper is based upon work supported by the Center for Minds, Brains and Machines (CBMM), funded by NSF STC award CCF-1231216 and by NSF award 213418.

References

J. M. Alvarez and M. Salzmann. Compression-aware training of deep networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 856–867, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

M. Anthony and P. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002.

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 10–15 Jul 2018.

L. Bottou. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*, Nimes, France, 1991. EC2. URL http://leon.bottou.org/papers/bottou-91c.

E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/2afe4567e1bf64d32a5527244d104cea-Paper.pdf.

H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Mach. Learn.*, 110(2):393–416, feb 2021. ISSN 0885-6125. doi: 10.1007/s10994-020-05929-w. URL https://doi.org/10.1007/s10994-020-05929-w.

P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf.

S. Gunasekar, B. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro. Implicit regularization in matrix factorization, 2017. URL https://arxiv.org/abs/1705.09280.

K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/a5e0ff62be0b08456fc7f1e88812af3d-Paper.pdf.

S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models, 2017. URL https://arxiv.org/abs/1702.03275.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

S. Jastrzębski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Three factors influencing minima in sgd, 2017. URL https://arxiv.org/abs/1711.04623.

Z. Ji and M. Telgarsky. Directional convergence and alignment in deep learning. *CoRR*, abs/2006.06657, 2020. URL https://arxiv.org/abs/2006.06657.

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

A. Krizhevsky. One weird trick for parallelizing convolutional neural networks, 2014.

Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 319–345. Springer Verlag, 1999. ISBN 3540667229. doi: 10.1007/3-540-46805-6_19. International Workshop on Shape, Contour and Grouping in Computer Vision ; Conference date: 26-05-1998 Through 29-05-1998.

Z. Li, Y. Luo, and K. Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *CoRR*, abs/2012.09839, 2020. URL https://arxiv.org/abs/2012.09839.

Z. Li, T. Wang, and S. Arora. What happens after SGD reaches zero loss? -a mathematical framework. *CoRR*, abs/2110.06914, 2021. URL https://arxiv.org/abs/2110.06914.

T. Poggio and Q. Liao. Generalization in deep network classifiers trained with the square loss. *CBMM Memo No. 112*, 2019.

T. Poggio, A. Banburski, and Q. Liao. Theoretical issues in deep networks. *PNAS*, 2020.

D. Povey, X. Zhang, and S. Khudanpur. Parallel training of dnns with natural gradient and parameter averaging, 2014. URL https://arxiv.org/abs/1410.7455.

T. Raiko, H. Valpola, and Y. Lecun. Deep learning made easier by linear transformations in perceptrons. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 924–932, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL https://proceedings.mlr.press/v22/raiko12.html.

N. Razin and N. Cohen. Implicit regularization in deep learning may not be explainable by norms. *CoRR*, abs/2005.06398, 2020. URL https://arxiv.org/abs/2005.06398.

N. Timor, G. Vardi, and O. Shamir. Implicit regularization towards rank minimization in relu networks. *CoRR*, abs/2201.12760, 2022. URL https://arxiv.org/abs/2201.12760.

M. Tukan, A. Maalouf, M. Weksler, and D. Feldman. No fine-tuning, no cry: Robust svd for compressing deep networks. *Sensors*, 21(16), 2021. ISSN 1424-8220. doi: 10.3390/s21165599. URL https://www.mdpi.com/1424-8220/21/16/5599.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

S. Wiesler, A. Richard, R. Schlüter, and H. Ney. Mean-normalized stochastic gradient for large-scale deep learning. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 180–184, 2014.

J. Xu, X. Sun, Z. Zhang, G. Zhao, and J. Lin. Understanding and improving layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/2f4fe03d77724a7217006e5d16728874-Paper.pdf.

X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 67–76, 2017. doi: 10.1109/CVPR.2017.15.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.

Z. Zhu, J. Wu, B. Yu, L. Wu, and J. Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? **[TODO]**
   (b) Did you include complete proofs of all theoretical results? **[TODO]**
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)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
   (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)? **[TODO]**
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? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**
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? **[TODO]**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**

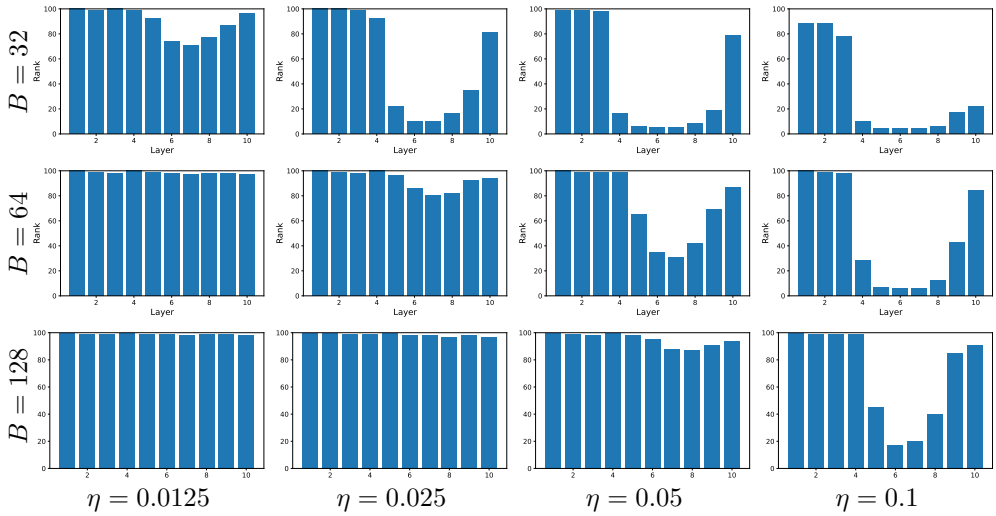


Figure 6: **Ranks across layers of MLP-BN-10-100 trained on CIFAR10 with varying learning rates and batch sizes.** The x-axis specifies the layer's index and the y-axis stands for the rank of the given matrix. As can be seen, MLPs are inductively biased towards learning a bottleneck.

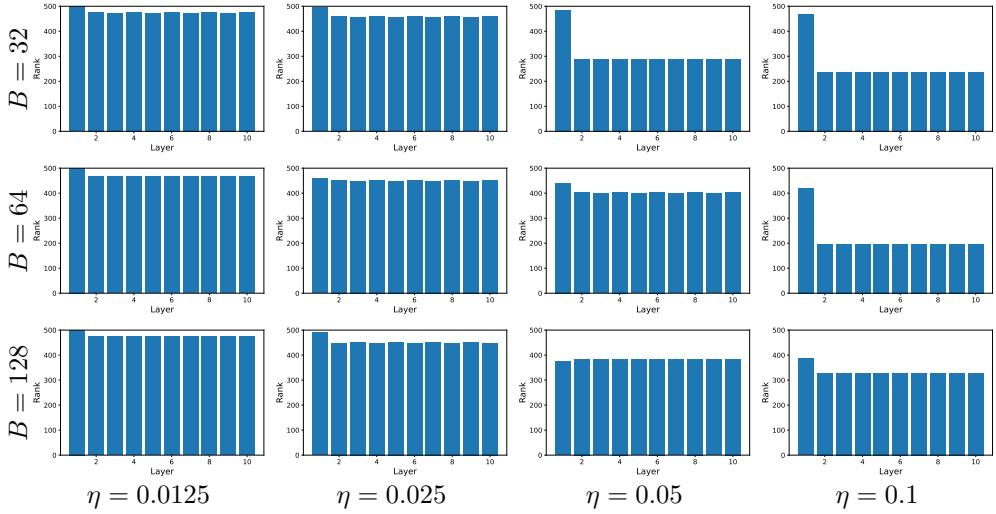


Figure 7: **Ranks across layers of RES-5-500 trained on CIFAR10 with varying learning rates and batch sizes.** The x-axis specifies the layer's index and the y-axis stands for the rank of the given matrix. As can be seen, ResNets are inductively biased to balance the rank of their residual blocks.

A Additional Experiments

Inductive biases. As an additional experiment we look at the rank of each matrix learned by MLP-BN-10-100 and RES-BN-5-500. We train each model with a different batch size $B \in \{32, 64, 128\}$ and initial learning rate $\eta \in \{0.0125, 0.025, 0.05, 0.1\}$, while fixing $\lambda = 5\text{e}{-4}$ and the momentum to be 0.9. In Fig. 6 we plot the ranks of each one of the matrices of the MLP-BN-10-100 trained models and in Fig. 7 we plot the ranks for the RES-BN-5-500 models. Similar to Figs. 2 and 4, when the batch sizes are too large or the learning rates are too small, the low-rank constraint is negligible. However, when the rank constraint is active, we observe interesting inductive biases that emerge when training the architectures. Specifically, for MLPs we obtain an encoder-decoder structure in which the ranks of the bottom and top layers are larger then the ranks of the mid-layers. In contrast, for ResNets we obtain a balanced structure in which the matrices within the residual blocks are of the same rank.

B Analyzing Standard Neural Networks

Lemma 1. *Let f_W be a neural network and W^{pq} be a given matrix within f_W. Then, for any input $x \in \mathbb{R}^n$, we have* $\text{rank}\left(\frac{\partial f_W(x)}{\partial W^{pq}}\right) \leq 1$.

Proof. We would like to show that the matrix $\frac{\partial f_W(x)}{\partial W^{pq}}$ is of rank ≤ 1. We note that for any fixed x, the network can be written as follows,

$$f_W(x) \;=\; \sum_{l_1 \in \text{pred}(l_0)} W^{l_0 l_1} \cdot v_{l_1}(x), \tag{11}$$

where $l_0 = L$. In addition, each layer v_l can also be written as

$$v_{l_1}(x) \;=\; D_{l_1}(x) \sum_{l_2 \in \text{pred}(l_1)} W^{l_1 l_2} \cdot v_{l_2}(x), \tag{12}$$

where $D_l(x) = \text{diag}[\sigma'(v_l(x))]$. Therefore, for any fixed input x, we can write $f(x)$ as the sum of matrix multiplications along paths π from v_1 to v_{l_0}. Specifically, we can write $f(x)$ as a follows

$$\begin{aligned}
f_W(x) \;=\; & \sum_{\pi \text{ from } p \text{ to } l_0} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x) \cdots D_{\pi_2}(x) \cdot W^{\pi_2 \pi_1} \cdot D_{\pi_1}(x) \cdot W^{pq} \cdot v_q(x) \\
& + \sum_{\substack{\pi \text{ from } 1 \text{ to } l_0 \\ (p,q) \notin \pi}} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x) \cdot W^{\pi_{T-1} \pi_{T-2}} \cdots D_{\pi_2}(x) \cdot W^{\pi_2 \pi_1} x \\
= \; & \left[\sum_{\pi \text{ from } p \text{ to } l_0} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x) \cdots W^{\pi_2 \pi_1} \cdot D_{\pi_1}(x) \right] \cdot W^{pq} \cdot v_q(x) \\
& + \sum_{\substack{\pi \text{ from } 1 \text{ to } l_0 \\ (p,q) \notin \pi}} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x) \cdot W^{\pi_{T-1} \pi_{T-2}} \cdots D_{\pi_2}(x) \cdot W^{\pi_2 \pi_1} x \\
=: \; & u_q(x)^\top \cdot W^{pq} \cdot v_q(x) \\
& + \sum_{\substack{\pi \text{ from } 1 \text{ to } l_0 \\ (p,q) \notin \pi}} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x) \cdot W^{\pi_{T-1} \pi_{T-2}} \cdots D_{\pi_2}(x) \cdot W^{\pi_2 \pi_1} x,
\end{aligned}$$

where $T := T(\pi)$ denotes the length of the path π. We note that with measure 1 over W^{pq} and x, the matrices $D_p(x)$ are constant in the neighborhood of W^{pq}. In addition, $v_q(x)$ is independent of W^{pq}. Therefore, we conclude that

$$\frac{\partial f_W(x)}{\partial W^{pq}} \;=\; u_q(x) \cdot v_q(x)^\top. \tag{13}$$

Since $v_q(x)$ and $u_q(x)$ are vectors, we conclude that $\text{rank}\left(\frac{\partial f_W(x)}{\partial W^{pq}}\right) \leq 1$. □

Proposition 1 ($\lambda > 0$)**.** *Let $\ell(a, b)$ be a differentiable loss function, $\lambda > 0$, and let $f_W(x)$ be a ReLU neural network, where $\{(v_p, v_1) \in E \mid v_p \in V\} \subset E_T$. Let W be a convergence point of mini-batch SGD for minimizing $L_S^\lambda(f_W)$. Then, either $f_W \equiv 0$ or $\{x_i\}_{i=1}^m$ are collinear vectors.*

Proof. By the same argument as in the proof of Lem. 1, we can write

$$\begin{aligned}
f_W(x) \;=\; & \sum_{\pi \text{ from } 1 \text{ to } L} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x_i) \cdot W^{\pi_{T-1} \pi_{T-2}} \cdots D_{\pi_2}(x_i) \cdot W^{\pi_2 \pi_1} \cdot x \\
= \; & \sum_{(v_p, v_1) \in E} E_p(x) \cdot W^{p1} \cdot x \\
= \; & \sum_{(v_p, v_1) \in E_T} E_p(x) \cdot W^{p1} \cdot x,
\end{aligned} \tag{14}$$

where $E_p(x) := \sum_{\pi \text{ from } p \text{ to } L} W^{\pi_T \pi_{T-1}} \cdot D_{\pi_{T-1}}(x_i) \cdots W^{\pi_2 \pi_1} \cdot D_{\pi_1}(x_i)$. We denote $E_{p,i} := E_p(x_i)$ and we can write, $\frac{\partial f_W(x_i)}{\partial W^{p1}} = E_{p,i} \cdot x_i^\top$.

We would like to show that $\{x_i\}_{i=1}^m$ are collinear vectors or $f_W \equiv 0$. Assume the opposite by contradiction, i.e., that $\{x_i\}_{i=1}^m$ are not collinear vectors and that $f_W \not\equiv 0$. In particular, by Eq. 14, we have, $W^{p1} \neq 0$ for some p.

According to the analysis in Sec. 3.1, $\{\frac{\partial f_W(x_i)}{\partial \text{vec}(W^{p1})}\}_{i=1}^m$. Therefore, for any pair $i, j \in [m]$, there is a scalar $\alpha_{ij}^p \in \mathbb{R}$, such that, $E_{p,i} \cdot x_i^\top = \alpha_{ij}^p E_{p,j} \cdot x_j^\top$.

Consider a given index $i \in [m]$. We would like to show that either $E_{p,i} = 0$ or $\{x_i\}_{i=1}^m$ are collinear vectors. For any $j \in [m]$, if $\alpha_{ij}^p = 0$, then, $E_{p,i} \cdot x_i^\top = 0$, which implies that either $E_{p,i} = 0$ or $x_i = 0$ (which is false by assumption). Otherwise ($\alpha_{ij}^p \neq 0$), either $E_{p,i} = 0$ or x_i and x_j are collinear. We note that since this argument holds for all $j \in [m]$, unless $E_{p,i} = 0$, we obtain that $\{x_i, x_j\}$ are collinear for all $j \in [m]$. Note that if $\{x_i, x_j\}$ are collinear for all $j \in [m]$, then, $\{x_i\}_{i=1}^m$ are collinear vectors. Therefore, we state that either $E_{p,i} = 0$ or $\{x_i\}_{i=1}^m$ are collinear vectors (which we assumed to be false). We note that if $E_{p,i} = 0$ for all $i \in [m]$, then, $M^{p1} = 0$ (see Eq. 8), and by Eq. 9, we have, $W^{p1} = 0$ in contradiction. Therefore, we conclude that either $\{x_i\}_{i=1}^m$ are collinear vectors or $f_W \equiv 0$. □

Proposition 2 ($\lambda = 0$)**.** *Let $\ell(a, b)$ be a differentiable loss function, $\lambda = 0$, and let $f_W(x)$ be a ReLU neural network, where $\{(v_p, v_1) \in E \mid v_p \in V\} \subset E_T$. Let W be a convergence point of mini-batch SGD for minimizing $L_S^\lambda(f_W)$. Assume that $\{x_i\}_{i=1}^m$ are not collinear. Then, for any $i \in [m]$, for which $\frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \neq 0$, we have, $f_W(x_i) = 0$. In particular, if ℓ is convex and $\forall\, b \in \mathbb{R} \nexists\, a^* \in \mathbb{R} :\ \ell(a^*, b) = \inf_a \ell(a, b)$, then, $\forall\, i \in [m] :\ f_W(x_i) = 0$.*

Proof. Let $i \in [m]$ be an index for which $\frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \neq 0$. Then, we consider two possibilities: (i) there exist $j \in [m]$, for which $\frac{\partial \ell(f_W(x_j), y_j)}{\partial f_W(x_j)} = 0$ or (ii) for all $j \in [m]$, we have $\frac{\partial \ell(f_W(x_j), y_j)}{\partial f_W(x_j)} \neq 0$. Assume that the former holds. Then, by Eq. 8, we have,

$$\forall\, (v_p, v_1) \in E :\ \frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \cdot \frac{\partial f_W(x_i)}{\partial W^{p1}} \ = \ 0. \tag{15}$$

This implies that $E_{p,i} \cdot x_i^\top = \frac{\partial f_W(x_i)}{\partial W^{p1}} = 0$ for all $(v_p, v_1) \in E$ (see the proof of Prop. 1). Since $x_i \neq 0$, we conclude that $E_{p,i} = 0$ for all $(v_p, v_1) \in E$, which implies that $f_W(x_i) = 0$ by Eq. 14.

If the latter holds, then, for all $i, j \in [m]$ and trainable W^{pq}, we have

$$\frac{\partial f_W(x_i)}{\partial W^{pq}} \ = \ \left[\frac{\partial \ell(f_W(x_j), y_j)}{\partial f_W(x_j)}\right] / \left[\frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)}\right]^{-1} \cdot \frac{\partial f_W(x_j)}{\partial W^{pq}}. \tag{16}$$

In particular, $\{\frac{\partial f_W(x_i)}{\partial \text{vec}(W^{p1})}\}_{i=1}^m$ are collinear for all $(v_p, v_1) \in E$. Hence, by the proof of Cor. 1, either $\forall\, i \in [m]\ \forall\, (v_p, v_1) \in E :\ E_{p,i} = 0$ or $\{x_i\}_{i=1}^m$ are collinear. Therefore, since $\{x_i\}_{i=1}^m$ are not collinear, we obtain that $f_W(x_i) = 0$ for all $i \in [m]$ by Eq. 14.

Finally, if ℓ is convex and $\forall\, b \in \mathbb{R} \nexists\, a^* \in \mathbb{R} :\ \ell(a^*, b) = \inf_a \ell(a, b)$, we obtain that $\forall\, i \in [m] :\ \frac{\partial \ell(f_W(x_i), y_i)}{\partial f_W(x_i)} \neq 0$. Hence, by the above, we conclude that, $\forall\, i \in [m] :\ f_W(x_i) = 0$. □

Theorem 1. *Let $\|\cdot\|$ be any matrix norm. Let $f_W(x, \tilde{\mathcal{X}})$ be a ReLU neural network and W^{pq} be a given matrix within f_W and let $B^* < m$ be a natural number. Then,*

$$\min_{\substack{V \in R^{d_p \times d_q}: \\ \text{rank}(V) \leq B^*}} \|W^{pq} - V\| \ \leq \ \frac{1}{2\lambda} \min_{\tilde{S} \subset S:\ |\tilde{S}| = B^*} \|\nabla_{W^{pq}} L_{\tilde{S}}^\lambda(\tilde{f}_W)\| \tag{10}$$

Proof. Let $\tilde{S} = \{(x_{ij}, y_{ij})\}_{j=1}^{B^*} \subset S$ be a batch of B^* samples and $\tilde{\mathcal{X}} = \{x_{ij}\}_{j=1}^{B^*}$ be the corresponding unlabeled batch. By the chain rule, we can write the gradient of the loss function as follows

$$\frac{\bar{\partial} L_{\tilde{S}}(\tilde{f}_W)}{\partial W^{pq}} \ = \ \frac{1}{B^*} \sum_{j=1}^{B^*} \frac{\partial \ell(f_W(x_{ij}; \tilde{\mathcal{X}}), y_{ij})}{\partial f_W(x_{ij}; \tilde{\mathcal{X}})} \cdot \frac{\bar{\partial} f_W(x_{ij}; \tilde{\mathcal{X}})}{\partial W^{pq}} + 2\lambda W^{pq}. \tag{17}$$

According to Lems. 1 and 2, the matrix $U_{\tilde{S}} := -\frac{1}{2\lambda B^*}\sum_{j=1}^{B^*}\frac{\ell(f_W(x_{ij};\tilde{\mathcal{X}}),y_{ij})}{\partial f_W(x_{ij};\tilde{\mathcal{X}})}\cdot\frac{\partial f_W(x_{ij};\tilde{\mathcal{X}})}{\partial W^{pq}}$ is of rank $\leq B^*$. Therefore, we obtain the following

$$\begin{aligned}\min_{V:\ \text{rank}(V)\leq B^*}\|W^{pq}-V\| \ &\leq \ \min_{\tilde{S}\subset S:\ |\tilde{S}|=B^*}\|W^{pq}-U_{\tilde{S}}\| \\ &= \ \frac{1}{2\lambda}\min_{\tilde{S}\subset S:\ |\tilde{S}|=B^*}\|\nabla_{W^{pq}}L_{\tilde{S}}(\tilde{f}_W)\|,\end{aligned} \tag{18}$$

which completes the proof of this theorem. □

C Analyzing Neural Networks with Batch Normalization

Lemma 2. *Let $\tilde{f}_W$ be a neural network and W^{pq} be a given matrix within f_W. Then, for any batch $\tilde{\mathcal{X}}=\{x_j\}_{j=1}^B$ and real values $\beta_1,\ldots,\beta_B\in\mathbb{R}$, we have* $\text{rank}\left(\sum_{j=1}^B\beta_j\cdot\frac{\partial\tilde{f}_W(x_j;\tilde{\mathcal{X}})}{\partial W^{pq}}\right)\leq B$.

Proof. We would like to show that the matrices $\frac{\bar{\partial}f_W(x_i;\tilde{\mathcal{X}})}{\partial W^{pq}}$ and $\sum_{i=1}^B\beta_i\cdot\frac{\bar{\partial}f_W(x_i;\tilde{\mathcal{X}})}{\partial W^{pq}}$ are of rank $\leq B$. We note that for any fixed x, the network can be written as follows

$$f(x;\tilde{\mathcal{X}}) \ = \ \sum_{l_1\in\text{pred}(L)}W^{Ll_1}\cdot v_{l_1}(x;\tilde{\mathcal{X}}). \tag{19}$$

In addition, each layer v_{l_1} can also be written as

$$\begin{aligned}v_{l_1}(x) \ &= \ D_{l_1}(x;\tilde{\mathcal{X}})\cdot\left[\gamma_{l_1}\cdot A_{l_1}(\tilde{\mathcal{X}})\cdot\left(\sum_{l_2\in\text{pred}(l_1)}W^{l_1l_2}\cdot v_{l_2}(x;\tilde{\mathcal{X}})-b_{l_2}(\tilde{\mathcal{X}})\right)+\mathbb{1}\cdot\beta_{l_1}\right] \\ &= \ D_{l_1}(x;\tilde{\mathcal{X}})\cdot\left[\gamma_{l_1}\cdot A_{l_1}(\tilde{\mathcal{X}})\cdot\left(\sum_{l_2\in\text{pred}(l_1)}W^{l_1l_2}\cdot\left[v_{l_2}(x;\tilde{\mathcal{X}})-\mu_{l_2}(\tilde{\mathcal{X}})\right]\right)+\mathbb{1}\cdot\beta_{l_1}\right] \\ &= \ \sum_{l_2\in\text{pred}(l_1)}Q_{l_1}(x;\tilde{\mathcal{X}})\cdot W^{l_1l_2}\cdot\Big(v_2(x;\tilde{\mathcal{X}})-\mu_2(\tilde{\mathcal{X}})\Big)+D_{l_1}(x;\tilde{\mathcal{X}})\cdot\mathbb{1}\cdot\beta_{l_1}\end{aligned}$$

where $D_l(x;\tilde{\mathcal{X}}):=\text{diag}[\sigma'(n_l(v_l(x;\tilde{\mathcal{X}})\tilde{\mathcal{X}}))]$, $b_l(\tilde{\mathcal{X}}):=\sum_{t\in\text{pred}(l)}W^{lt}\cdot\mu_t(\tilde{\mathcal{X}}):=\sum_{t\in\text{pred}(l)}W^{lt}\cdot\mathbb{E}_{x\sim U[\tilde{\mathcal{X}}]}\left[v_t(x;\tilde{\mathcal{X}})\right]$ and $Q_l(x;\tilde{\mathcal{X}}):=\gamma_l\cdot D_l(x;\tilde{\mathcal{X}})\cdot A_l(x;\tilde{\mathcal{X}})$. For simplicity, we write $l_0:=L$, $\mu_l:=\mu_l(x;\tilde{\mathcal{X}})$, $v_l^j:=v_l(x_j;\tilde{\mathcal{X}})$ and $Q_l^j:=Q_l(x_j;\tilde{\mathcal{X}})$. We define

$$\Delta_l^j:=\begin{cases}x_j & \text{if } l=1\\ v_l^j-\mu_l & \text{if } l\neq 1\end{cases} \tag{20}$$

Hence, we can write $f(x_{j_0};\tilde{\mathcal{X}})$ as a follows

$$\begin{aligned}
f(x_{j_0};\tilde{\mathcal{X}}) &= \sum_{l_1\in\text{pred}(l_0)} W^{l_0 l_1}\cdot v_{l_1}^{j_0} \\
&= \sum_{l_1\in\text{pred}(l_0)} W^{l_0 l_1}\cdot v_{l_1}^{j_0} \\
&= \sum_{l_1\in\text{pred}(l_0)} W^{l_0 l_1}\cdot\left[Q_{l_1}^{j_0}\cdot\left(\sum_{l_2\in\text{pred}(l_1)} W^{l_1 l_2}\cdot(v_{l_2}^{j_0}-\mu_{l_2})\right)+\mathbb{1}\cdot\beta_{l_1}\right] \\
&= \sum_{l_1\in\text{pred}(l_0)} W^{l_0 l_1}\cdot\mathbb{1}\cdot\beta_{l_1} \\
&\quad+\sum_{l_1\in\text{pred}(l_0)}\sum_{l_2\in\text{pred}(l_1)} W^{l_0 l_1}\cdot Q_{l_1}^{j_0}\cdot W^{l_1 l_2}\cdot(v_{l_2}^{j_0}-\mu_{l_2}) \\
&= \sum_{l_1\in\text{pred}(l_0)} W^{l_0 l_1}\cdot\mathbb{1}\cdot\beta_{l_1} \\
&\quad+\sum_{l_1\in\text{pred}(l_0)}\sum_{l_2\in\text{pred}(l_1)} W^{l_0 l_1}\cdot Q_{l_1}^{j_0}\cdot W^{l_1 l_2}\cdot\Delta_{l_2}^{j_0}.
\end{aligned} \tag{21}$$

For any $l_{t+2}\neq 1$, we can write

$$\begin{aligned}
\Delta_{l_{t+2}}^{j_t} &= v_{l_{t+2}}^{j_t}-\mu_{l_{t+2}} \\
&= D_{l_{t+2}}^{j_t}\cdot\mathbb{1}\cdot\beta_{l_{t+2}} \\
&\quad+Q_{l_{t+2}}^{j_t}\cdot\left(\sum_{l_{t+3}\in\text{pred}(l_{t+2})} W^{l_{t+2}l_{t+3}}\cdot(v_{l_{t+3}}^{j_t}-\mu_{l_{t+3}})\right) \\
&\quad-\frac{1}{B}\sum_{j_{t+1}=1}^{B} Q_{l_{t+2}}^{j_{t+1}}\cdot\left(\sum_{l_{t+2}\in\text{pred}(l_{t+1})} W^{l_{t+1}l_{t+2}}\cdot(v_{l_{t+2}}^{j_t}-\mu_{l_{t+2}})\right) \\
&= D_{l_{t+2}}^{j_t}\cdot\mathbb{1}\cdot\beta_{l_{t+2}}+\sum_{l_{t+3}\in\text{pred}(l_{t+2})}\sum_{j_{t+1}=1}^{B}\alpha_{j_t,j_{t+1}}Q_{l_{t+2}}^{j_{t+1}}\cdot W^{l_{t+2}l_{t+3}}\cdot\Delta_{l_{t+3}}^{j_{t+1}},
\end{aligned} \tag{22}$$

where $\alpha_{i,j}=-1/B$ for all pairs $i\neq j$ and $\alpha_{ii}=\frac{B-1}{B}$ otherwise.

Hence, by recursion we can write the T'th expansion of f_W,

$$\begin{aligned}
f_W(x_{j_0};\tilde{\mathcal{X}}) &= \sum_{l_1\in\text{pred}(l_0)} W^{l_0 l_1}\cdot\mathbb{1}\cdot\beta_{l_1} \\
&\quad+\sum_{l_1\in\text{pred}(l_0)}\sum_{l_2\in\text{pred}(l_1)} W^{l_0 l_1}\cdot Q_{l_1}^{j_0}\cdot W^{l_1 l_2}\cdot D_{l_2}^{j_0}\cdot\mathbb{1}\cdot\beta_{l_2} \\
&\quad+\sum_{t=2}^{T}\sum_{l_1\in\text{pred}(l_0)}\sum_{l_2\in\text{pred}(l_1)}\sum_{l_3\in\text{pred}(l_2)}\cdots\sum_{l_{t+1}\in\text{pred}(l_t)} W^{l_0 l_1}\cdot Q_{l_1}^{j_0} \\
&\qquad\cdot W^{l_1 l_2}\cdot\sum_{j_1=1}^{B}\alpha_{j_0,j_1}Q_{l_2}^{j_1}\cdots W^{l_t l_{t+1}}\cdot\sum_{j_t=1}^{B}\alpha_{j_{t-1},j_t}Q_{l_{t+1}}^{j_t}\cdot D_{l_t}^{j_{t-1}}\cdot\mathbb{1}\cdot\beta_{l_t} \\
&\quad+\sum_{l_1\in\text{pred}(l_0)}\sum_{l_2\in\text{pred}(l_1)}\sum_{l_3\in\text{pred}(l_2)}\cdots\sum_{l_{t+1}\in\text{pred}(l_T)} W^{l_0 l_1}\cdot Q_{l_1}^{j_0} \\
&\qquad\cdot W^{l_1 l_2}\cdot\sum_{j_1=1}^{B}\alpha_{j_0,j_1}Q_{l_2}^{j_1}\cdots W^{l_T l_{T+1}}\cdot\sum_{j_T=1}^{B}\alpha_{j_{T-1},j_T}Q_{l_{T+1}}^{j_T}\cdot W^{l_{T+1}l_{T+2}}\cdot\Delta_{l_{T+2}}^{j_T}.
\end{aligned}$$

Instead of recursively expanding f_W's expression T times by rewriting $\Delta_{l_{T+1}}^{j_T}$ using Eq. 22, we can expand through $\Delta_{l_{T+2}}^{j_T}$ based on different kinds of criteria. Specifically, we can keep $\Delta_{l_{T+2}}^{j_T}$ unchanged or expanded based on a whether $(l_T, l_{T+1}) = (p, q)$ or not. Therefore, we can write $f_W(x_{j_0}; \tilde{\mathcal{X}})$ as the sum of matrix multiplications along paths π that start with (q, p) and end by L and paths that start with 1 and end with L that do not pass through (q, p). Formally, let $j_0 \in [B]$ and π a path from some node to $\pi_T = L$ of length T,

$$E_{1,\pi}^{j_0} = \begin{cases} W^{\pi_2\pi_1} \cdot \mathbb{1} \cdot \beta_{\pi_1} & \text{if } T = 2 \\ W^{\pi_3\pi_2} \cdot Q_{\pi_2}^{j_0} \cdot W^{\pi_2\pi_1} \cdot D_{\pi_1}^{j_0} \cdot \mathbb{1} \cdot \beta_{\pi_1} & \text{if } T = 3 \\ W^{\pi_T\pi_{T-1}} \cdot Q_{\pi_{T-1}}^{j_0} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q_{\pi_{T-2}}^{j_1} & \\ \quad \cdots W^{\pi_2\pi_1} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q_{\pi_1}^{j_{T-2}} \cdot D_{\pi_1}^{j_{T-2}} \cdot \mathbb{1} \cdot \beta_{\pi_1} & \text{if } T > 3 \end{cases}$$

and also

$$E_{2,\pi}^{j_0} = \begin{cases} W^{\pi_T\pi_{T-1}} \cdot Q_{\pi_{T-1}}^{j_0} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q_{\pi_{T-2}}^{j_1} & \\ \quad \cdots W^{\pi_2\pi_1} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q_{\pi_1}^{j_{T-2}} \cdot \sum_{l \in \text{pred}(\pi_1)} W^{\pi_1 l} \cdot \Delta_l^{j_{T-2}} & \text{if } \pi_1 \neq 1 \\ W^{\pi_T\pi_{T-1}} \cdot Q_{\pi_{T-1}} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q_{\pi_{T-2}}^{j_1} & \\ \quad \cdots W^{\pi_3\pi_2} \cdot \sum_{j_t=1}^{B} \alpha_{j_{t-1},j_t} Q_{\pi_2}^{j_t} \cdot W^{\pi_2\pi_1} \cdot x & \text{if } \pi_1 = 1 \end{cases}$$

With this terminology we can write

$$\begin{aligned} f_W(x_{j_0}; \tilde{\mathcal{X}}) \; &= \sum_{\substack{\pi \text{ from some node to } L \\ (p,\, q) \notin \pi}} E_{1,\pi}^{j_0} + \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1 = p, \pi_2 = q}} E_{1,\pi}^{j_0} + \sum_{\substack{\pi \text{ from 1 to } L \\ (p,\, q) \notin \pi}} E_{2,\pi}^{j_0} + \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1 = p, \pi_2 = q}} E_{2,\pi}^{j_0} \\ &=: \; Z_1^{j_0} + Z_2^{j_0} + Z_3^{j_0} + Z_4^{j_0}. \end{aligned}$$

We note that with measure 1 over W, x and $\tilde{\mathcal{X}}$, the matrices $D_l(x; \tilde{\mathcal{X}})$ are constant in the neighborhood of W^{pq}. Since $A_l(x; \tilde{\mathcal{X}})$ is detached, the pseudo-gradients of the matrices Q_l^j with respect to W^{pq} are zero. Therefore, since the terms $Z_1^{j_0}$ and $Z_3^{j_0}$ do not involve W^{pq}, we have $\frac{\tilde{\partial} f_W(x_{j_0}; \tilde{\mathcal{X}})}{\partial W^{pq}} = \frac{\tilde{\partial}(Z_2^{j_0} + Z_4^{j_0})}{\partial W^{pq}}$.

As a next step, we would like to represent the pseudo gradients $\frac{\bar{\partial} Z_2}{\partial W^{pq}}$ and $\frac{\bar{\partial} Z_4}{\partial W^{pq}}$ individually. We start with $\frac{\bar{\partial} Z_2}{\partial W^{pq}}$. We can write

$$\begin{aligned}
Z_2^{j_0} &= \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1=p, \pi_2=q}} W^{\pi_T \pi_{T-1}} \cdot Q^{j_0}_{\pi_{T-1}} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q^{j_1}_{\pi_{T-2}} \\
&\qquad \cdots W^{\pi_2 \pi_1} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q^{j_{T-2}}_{\pi_1} \cdot D^{j_{T-2}}_{\pi_1} \cdot \mathbb{1} \cdot \beta_{\pi_1} \\
&= \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1=p, \pi_2=q}} \sum_{j_{T-3}=1}^{B} (u^{\pi,1}_{j_0,j_{T-3}})^\top \cdot W^{pq} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q^{j_{T-2}}_{q} \cdot D^{j_{T-2}}_{q} \cdot \mathbb{1} \cdot \beta_q \\
&= \sum_{j_{T-3}=1}^{B} \left[\sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1=p, \pi_2=q}} (u^{\pi,1}_{j_0,j_{T-3}})^\top \right] \cdot W^{pq} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q^{j_{T-2}}_{q} \cdot D^{j_{T-2}}_{q} \cdot \mathbb{1} \cdot \beta_q \\
&= \sum_{j_{T-3}=1}^{B} (u^{1}_{j_0,j_{T-3}})^\top \cdot W^{pq} \cdot u^{2}_{j_{T-3}},
\end{aligned}$$

where

$$\begin{aligned}
(u^{\pi,1}_{j_0,j_{T-3}})^\top &:= W^{\pi_T \pi_{T-1}} \cdot Q^{j_0}_{\pi_{T-1}} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q^{j_1}_{\pi_{T-2}} \\
&\qquad \cdots W^{\pi_4 \pi_3} \cdot \sum_{j_{T-4}=1}^{B} \alpha_{j_{T-5},j_{T-4}} Q^{j_{T-4}}_{\pi_3} \cdot W^{\pi_3 \pi_2} \alpha_{j_{T-4} j_{T-3}} Q^{j_{T-3}}_{\pi_2} \\
(u^{1}_{j_0,j_{T-3}})^\top &:= \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1=p, \pi_2=q}} u^{\pi,1}_{j_{T-3}} \\
u^{2}_{j_{T-3}} &:= \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q^{j_{T-2}}_{q} \cdot D^{j_{T-2}}_{q} \cdot \mathbb{1} \cdot \beta_q.
\end{aligned}$$

Therefore, we can write $Z_2^{j_0} = \sum_{j_{T-3}=1}^{B} (u^{1}_{j_0,j_{T-3}})^\top \cdot W^{pq} \cdot u^{2}_{j_{T-3}}$ and we obtain that $\frac{\bar{\partial} Z_2^{j_0}}{\partial W^{pq}} = \sum_{j_{T-3}=1}^{B} (u^{1}_{j_0,j_{T-3}}) \cdot (u^{2}_{j_{T-3}})^\top$.

Next, we would like to analyze the pseudo gradient $\frac{\bar{\partial} Z_4^{j_0}}{\partial W^{pq}}$. We can write

$$\begin{aligned}
Z_4^{j_0} &= \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1=p, \pi_2=q}} W^{\pi_T \pi_{T-1}} \cdot Q^{j_0}_{\pi_{T-1}} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q^{j_1}_{\pi_{T-2}} \\
&\qquad \cdots W^{\pi_2 \pi_1} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q^{j_{T-2}}_{\pi_1} \cdot \sum_{l \in \text{pred}(\pi_1)} W^{\pi_1 l} \cdot \Delta_l^{j_{T-2}} \\
&= \sum_{\substack{\pi \text{ from } p \text{ to } L \\ \pi_1=p, \pi_2=q}} W^{\pi_T \pi_{T-1}} \cdot Q^{j_0}_{\pi_{T-1}} \cdot W^{\pi_{T-1}\pi_{T-2}} \cdot \sum_{j_1=1}^{B} \alpha_{j_0,j_1} Q^{j_1}_{\pi_{T-2}} \\
&\qquad \cdots W^{pq} \cdot \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q^{j_{T-2}}_{q} \cdot \sum_{l \in \text{pred}(q)} W^{ql} \cdot \Delta_l^{j_{T-2}} \\
&= \sum_{j_{T-3}=1}^{B} (u^{1}_{j_0,j_{T-3}})^\top \cdot W^{pq} \cdot u^{3}_{j_{T-3}},
\end{aligned}$$

where

$$u^3_{j_{T-3}} := \sum_{j_{T-2}=1}^{B} \alpha_{j_{T-3},j_{T-2}} Q_q^{j_{T-2}} \cdot \sum_{l \in \text{pred}(q)} W^{ql} \cdot \Delta_l^{j_{T-2}}.$$

Therefore, we conclude that $\frac{\bar{\partial} Z_4^{j_0}}{\partial W^{pq}} = \sum_{j_{T-3}=1}^{B} (u^1_{j_0,j_{T-3}}) \cdot (u^3_{j_{T-3}})^\top$. Hence, we conclude that

$$\begin{aligned} \frac{\bar{\partial} f(x_{j_0}; \tilde{\mathcal{X}})}{\partial W^{pq}} &= \frac{\bar{\partial} Z_2^{j_0}}{\partial W^{pq}} + \frac{\bar{\partial} Z_4^{j_0}}{\partial W^{pq}} \\ &= \sum_{j_{T-3}=1}^{B} (u^1_{j_0,j_{T-3}}) \cdot (u^2_{j_{T-3}})^\top + \sum_{j_{T-3}=1}^{B} (u^1_{j_0,j_{T-3}}) \cdot (u^3_{j_{T-3}})^\top \\ &= \sum_{j_{T-3}=1}^{B} (u^1_{j_0,j_{T-3}}) \cdot (u^2_{j_{T-3}} + u^3_{j_{T-3}})^\top, \end{aligned} \tag{23}$$

which is a matrix of rank $\leq B$. Finally, we notice that

$$\begin{aligned} \sum_{j_0=1}^{B} \beta_{j_0} \cdot \frac{\bar{\partial} f_W(x_{j_0}; \tilde{\mathcal{X}})}{\partial W^{pq}} &= \sum_{j_0=1}^{B} \beta_{j_0} \left(\frac{\bar{\partial} Z_2^{j_0}}{\partial W^{pq}} + \frac{\bar{\partial} Z_4^{j_0}}{\partial W^{pq}} \right) \\ &= \sum_{j_0=1}^{B} \beta_{j_0} \cdot \sum_{j_{T-3}=1}^{B} (u^1_{j_0,j_{T-3}}) \cdot (u^2_{j_{T-3}} + u^3_{j_{T-3}})^\top \\ &= \sum_{j_{T-3}=1}^{B} \left(\sum_{j_0=1}^{B} \beta_{j_0} u^1_{j_0,j_{T-3}} \right) \cdot (u^2_{j_{T-3}} + u^3_{j_{T-3}})^\top, \end{aligned} \tag{24}$$

which is also a matrix of rank $\leq B$. □

D Generalization

Our estimate of the number of units and parameters needed for a deep network to approximate compositional functions with an error ϵ_G allow the use of one of several available bounds for the generalization error of the network to derive sample complexity bounds. It is important to notice however that these bounds *do not* apply to the networks used today, since they assume a number of parameters smaller than the size of the training set. We report them for the interest of the curious reader. Consider theorem 16.2 in Anthony and Bartlett [2002] which provides the following sample bound for a generalization error ϵ_G with probability at least $1 - \delta$ in a network in which the W parameters (weights and biases) which are supposed to minimize the empirical error (the theorem is stated in the standard ERM setup) are expressed in terms of k bits:

$$M(\epsilon_G, \delta) \leq \frac{2}{\epsilon_G^2}(kW \log 2 + \log(\frac{2}{\delta})) \tag{25}$$

E Rank one Constraint for Exponential-Type Loss Functions

Assume that a deep multilayered neural network $f(x) = W^L \sigma(W^{L-1} \dots \sigma(W^1 x))$ is trained on a binary classification task with weight normalization with respect to an exponential loss function without weight decay. The stationary points of the SGD flow of the normalized weight matrices V_k are given (see Poggio and Liao [2019], Poggio et al. [2020]) for any finite $\rho = \prod_{k=1}^{L} \rho_k$, where $\rho_k = \|W_k\|_2$ by

$$\sum_{n}^{B} e^{-\rho y_n \hat{f}_n} y_n \left(\frac{\partial \hat{f}_n}{\partial V_k} - V_k \hat{f}_n \right) = 0, \tag{26}$$

where $\hat{f}_n$ is the scalar output of the ReLU network with normalized weight matrices when the input is x_n, y_n is the corresponding binary label and B is the size of the minibatches. For any large but finite value of ρ, the equation shows a similar rank one constraint on the V_k weight matrices.