Gradient Descent Finds Global Minima of Deep Neural Networks

Simon S. Du [*1] **Jason D. Lee** [*2] **Haochuan Li** [*3 4] **Liwei Wang** [*5 4] **Xiyu Zhai** [*6]

Abstract

Gradient descent finds a global minimum in training deep neural networks despite the objective function being non-convex. The current paper proves gradient descent achieves zero training loss in polynomial time for a deep over-parameterized neural network with residual connections (ResNet). Our analysis relies on the particular structure of the Gram matrix induced by the neural network architecture. This structure allows us to show the Gram matrix is stable throughout the training process and this stability implies the global optimality of the gradient descent algorithm. We further extend our analysis to deep residual convolutional neural networks and obtain a similar convergence result.

1. Introduction

One of the mysteries in deep learning is randomly initialized first-order methods like gradient descent achieve zero training loss, even if the labels are arbitrary (Zhang et al., 2016). Over-parameterization is widely believed to be the main reason for this phenomenon as only if the neural network has a sufficiently large capacity, it is possible for this neural network to fit all the training data. For example, Lu et al. (2017) proved that except for a measure zero set, all functions cannot be approximated by ReLU networks with a width less than the input dimension. In practice, many neural network architectures are highly over-parameterized. For example, Wide Residual Networks have 100x parameters than the number of training data (Zagoruyko & Komodakis, 2016).

[*]Equal contribution [1]Machine Learning Department, Carnegie Mellon University [2]Data Science and Operations Department, University of Southern California [3]School of Physics, Peking University [4]Center for Data Science, Peking University, Beijing Institute of Big Data Research [5]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University [6]Department of EECS, Massachusetts Institute of Technology. Correspondence to: Simon S. Du <ssdu@cs.cmu.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

The second mysterious phenomenon in training deep neural networks is "deeper networks are harder to train." To solve this problem, He et al. (2016) proposed the deep residual network (ResNet) architecture which enables randomly initialized first order method to train neural networks with an order of magnitude more layers. Theoretically, Hardt & Ma (2016) showed that residual links in linear networks prevent gradient vanishing in a large neighborhood of zero, but for neural networks with non-linear activations, the advantages of using residual connections are not well understood.

In this paper, we demystify these two mysterious phenomena. We consider the setting where there are n data points, and the neural network has H layers with width m. We focus on the least-squares loss and assume the activation function is Lipschitz and smooth. This assumption holds for many activation functions including the soft-plus and sigmoid. Our contributions are summarized below.

- As a warm-up, we first consider a fully-connected feedforward network. We show if $m = \Omega\left(\mathrm{poly}(n)2^{O(H)}\right)$[1], then randomly initialized gradient descent converges to zero training loss at a linear rate.
- Next, we consider the ResNet architecture. We show as long as $m = \Omega\left(\mathrm{poly}(n, H)\right)$, then randomly initialized gradient descent converges to zero training loss at a linear rate. Comparing with the first result, the dependence on the number of layers improves exponentially for ResNet. This theory demonstrates the advantage of using residual architectures.
- Lastly, we apply the same technique to analyze convolutional ResNet. We show if $m = \mathrm{poly}(n, p, H)$ where p is the number of patches, then randomly initialized gradient descent achieves zero training loss.

Our proof builds on two ideas from previous work on gradient descent for two-layer neural networks. First, we use the observation by (Li & Liang, 2018) that if the neural network is over-parameterized, every weight matrix is close to its initialization. Second, following (Du et al., 2018b),

[1]The precise polynomials and data-dependent parameters are stated in Section 5, 6, 7.

we analyze the dynamics of the predictions whose convergence is determined by the least eigenvalue of the Gram matrix induced by the neural network architecture and to lower bound the least eigenvalue, it is sufficient to bound the distance of each weight matrix from its initialization.

Different from these two works, in analyzing deep neural networks, we need to exploit more structural properties of deep neural networks and develop new techniques for analyzing both the initialization and gradient descent dynamics. In Section 4 we give an overview of our proof technique.

1.1. Organization

This paper is organized as follows. In Section 2, we discuss related works. In Section 3, we formally state the problem setup. In Section 4, we present our main analysis techniques. In Section 5, we give a warm-up result for the deep fully-connected neural network. In Section 6, we give our main result for the ResNet. In Section 7, we give our main result for the convolutional ResNet. We conclude in Section 8 and defer all proofs to the appendix.

2. Related Works

Recently, many works try to study the optimization problem in deep learning. Since optimizing a neural network is a non-convex problem, one approach is first to develop a general theory for a class of non-convex problems which satisfy desired geometric properties and then identify that the neural network optimization problem belongs to this class. One promising candidate class is the set of functions that satisfy: a) all local minima are global and b) there exists a negative curvature for every saddle point. For this function class, researchers have shown (perturbed) gradient descent (Jin et al., 2017; Ge et al., 2015; Lee et al., 2016; Du et al., 2017a) can find a global minimum. Many previous works thus try to study the optimization landscape of neural networks with different activation functions (Soudry & Hoffer, 2017; Safran & Shamir, 2018; 2016; Zhou & Liang, 2017; Freeman & Bruna, 2016; Hardt & Ma, 2016; Nguyen & Hein, 2017; Kawaguchi, 2016; Venturi et al., 2018; Soudry & Carmon, 2016; Du & Lee, 2018; Soltanolkotabi et al., 2018; Haeffele & Vidal, 2015). However, even for a three-layer linear network, there exists a saddle point that does not have a negative curvature (Kawaguchi, 2016), so it is unclear whether this geometry-based approach can be used to obtain the global convergence guarantee of first-order methods.

Another way to attack this problem is to study the dynamics of a specific algorithm for a specific neural network architecture. Our paper also belongs to this category. Many previous works put assumptions on the input distribution and assume the label is generated according to a planted neural network. Based on these assumptions, one can obtain global convergence of gradient descent for some shallow neural networks (Tian, 2017; Soltanolkotabi, 2017; Brutzkus & Globerson, 2017; Du et al., 2018a; Li & Yuan, 2017; Du et al., 2017b). Some local convergence results have also been proved (Zhong et al., 2017a;b; Zhang et al., 2018). In comparison, our paper does not try to recover the underlying neural network. Instead, we focus on minimizing the training loss and rigorously prove that randomly initialized gradient descent can achieve zero training loss.

The most related papers are (Li & Liang, 2018; Du et al., 2018b) who observed that when training an over-parametrized two-layer fully-connected neural network, the weights do not change a large amount, which we also use to show the stability of the Gram matrix. They used this observation to obtain the convergence rate of gradient descent on a two-layer over-parameterized neural network for the cross-entropy and least-squares loss. More recently, Allen-Zhu et al. (2018b) generalized ideas from (Li & Liang, 2018) to derive convergence rates of training recurrent neural networks.

Our work extends these previous results in several ways: a) we consider deep networks, b) we generalize to ResNet architectures, and c) we generalize to convolutional networks. To improve the width dependence m on sample size n, we utilize a smooth activation (e.g. smooth ReLU). For example, our results specialized to depth $H = 1$ improve upon (Du et al., 2018b) in the required amount of overparametrization from $m = \Omega\left(n^6\right)$ to $m = \Omega\left(n^4\right)$. See Theorem 5.1 for the precise statement.

Chizat & Bach (2018b) brought to our attention the paper of Jacot et al. (2018) which proved a similar weight stability phenomenon for deep networks, but only in the asymptotic setting of infinite-width networks and gradient flow run for a finite time. Jacot et al. (2018) do not establish the convergence of gradient flow to a global minimizer. In lieu of their results, our work can be viewed as a generalization of their result to: a) finite width, b) gradient descent as opposed to gradient flow, and c) convergence to a global minimizer.

Mei et al. (2018); Chizat & Bach (2018a); Sirignano & Spiliopoulos (2018); Rotskoff & Vanden-Eijnden (2018); Wei et al. (2018) used optimal transport theory to analyze gradient descent on over-parameterized models. However, their results are limited to two-layer neural networks and may require an exponential amount of over-parametrization.

Daniely (2017) developed the connection between deep neural networks with kernel methods and showed stochastic gradient descent can learn a function that is competitive with the best function in the conjugate kernel space of the network. Andoni et al. (2014) showed that gradient descent can

learn networks that are competitive with polynomial classifiers. However, these results do not imply gradient descent can find a global minimum for the empirical loss minimization problem. Our analysis of the Gram matrices at random initialization is closely related to prior work on the analysis of infinite-width networks as Gaussian Processes (Raghu et al., 2016; Matthews et al., 2018; Lee et al., 2017; Schoenholz et al., 2016). Since we require the initialization analysis for three distinct architectures (ResNet, feed-forward, and convolutional ResNet), we re-derive many of these prior results in a unified fashion in Appendix E.

Finally, in concurrent work, Allen-Zhu et al. (2018c) also analyze gradient descent on deep neural networks. The primary difference between the two papers is that we analyze general smooth activations, and Allen-Zhu et al. (2018c) develop specific analysis for ReLU activation. The two papers also differ significantly on their data assumptions. *We wish to emphasize a fair comparison is not possible due to the difference in setting and data assumptions. We view the two papers as complementary since they address different neural net architectures.*

For ResNet, the primary focus of this manuscript, the required width per layer for Allen-Zhu et al. (2018c) is $m \gtrsim n^{30}H^{30}\log^2\frac{1}{\epsilon}$ and for this paper's Theorem 6.1 is $m \gtrsim n^4H^2$.[2] Our paper requires a width m that does not depend on the desired accuracy ϵ. As a consequence, Theorem 6.1 guarantees the convergence of gradient descent to a global minimizer. The iteration complexity of Allen-Zhu et al. (2018c) is $T \gtrsim n^6H^2\log\frac{1}{\epsilon}$ and of Theorem 6.1 is $T \gtrsim n^2\log\frac{1}{\epsilon}$.

For fully-connected networks, Allen-Zhu et al. (2018c) requires width $m \gtrsim n^{30}H^{30}\log^2\frac{1}{\epsilon}$ and iteration complexity $T \gtrsim n^6H^2\log\frac{1}{\epsilon}$. Theorem 5.1 requires width $m \gtrsim n^4 2^{O(H)}$ and iteration complexity $T \gtrsim n^2 2^{O(H)}\log\frac{1}{\epsilon}$. The primary difference is for very deep fully-connected networks, Allen-Zhu et al. (2018c) has milder dependence on H, but worse dependence on n. Commonly used fully-connected networks such as VGG are not extremely deep ($H = 16$), yet the dataset size such as ImageNet ($n \sim 10^6$) is very large.

In a second concurrent work, Zou et al. (2018) also analyzed the convergence of gradient descent on fully-connected networks with ReLU activation. The emphasis is on different loss functions (e.g. hinge loss), so the results are not directly comparable. Both Zou et al. (2018) and Allen-Zhu et al. (2018c) train a subset of the layers, instead of all the layers as in this work, but also analyze stochastic gradient.

[2] In all comparisons, we ignore the polynomial dependency on data-dependent parameters which only depends on the input data and the activation function. The two papers use different measures and are not directly comparable.

3. Preliminaries

3.1. Notations

We Let $[n] = \{1, 2, \ldots, n\}$. We use $N(\mathbf{0}, \mathbf{I})$ to denote the standard Gaussian distribution. For a matrix $\mathbf{A}$, we use $\mathbf{A}_{ij}$ to denote its (i,j)-th entry. We will also use $\mathbf{A}_{i,:}$ to denote the i-th row vector of $\mathbf{A}$ and define $\mathbf{A}_{i,j:k} = (\mathbf{A}_{i,j}, \mathbf{A}_{i,j+1}, \cdots, \mathbf{A}_{i,k})$ as part of the vector. Similarly $\mathbf{A}_{:,i}$ is the i-th column vector and $\mathbf{A}_{j:k,i}$ is a part of i-th column vector. For a vector $\mathbf{v}$, we use $\|\mathbf{v}\|_2$ to denote the Euclidean norm. For a matrix $\mathbf{A}$ we use $\|\mathbf{A}\|_F$ to denote the Frobenius norm and $\|\mathbf{A}\|_2$ to denote the operator norm. If a matrix $\mathbf{A}$ is positive semi-definite, we use $\lambda_{\min}(\mathbf{A})$ to denote its smallest eigenvalue. We use $\langle\cdot,\cdot\rangle$ to denote the standard Euclidean inner product between two vectors or matrices. We let $O(\cdot)$ and $\Omega(\cdot)$ denote standard Big-O and Big-Omega notations, only hiding constants. In this paper we will use C and c to denote constants. The specific value can be different from line to line.

3.2. Activation Function

We use $\sigma(\cdot)$ to denote the activation function. In this paper we impose some technical conditions on the activation function. The guiding example is softplus: $\sigma(z) = \log(1+\exp(z))$.

Condition 3.1 (Lipschitz and Smooth)**.** *There exists a constant $c > 0$ such that $|\sigma(0)| \le c$ and for any $z, z' \in \mathbb{R}$,*

$$|\sigma(z) - \sigma(z')| \le c|z - z'|,$$
$$\text{and } |\sigma'(z) - \sigma'(z)| \le c|z - z'|.$$

These two conditions will be used to show the stability of the training process. Note for softplus both Lipschitz constant and smoothness constant are 1. In this paper, we view all activation function related parameters as constants.

Condition 3.2. *$\sigma(\cdot)$ is analytic and is not a polynomial function.*

This assumption is used to guarantee the positive-definiteness of certain Gram matrices which we will define later. Softplus function satisfies this assumption by definition.

3.3. Problem Setup

In this paper, we focus on the empirical risk minimization problem with the quadratic loss function

$$\min_\theta L(\theta) = \frac{1}{2}\sum_{i=1}^n (f(\theta, \mathbf{x}_i) - y_i)^2 \tag{1}$$

where $\{\mathbf{x}_i\}_{i=1}^n$ are the training inputs, $\{y_i\}_{i=1}^n$ are the labels, θ is the parameter we optimize over and f is the prediction

function, which in our case is a neural network. We consider the following architectures.

- **Multilayer fully-connected neural networks:** Let $\mathbf{x} \in \mathbb{R}^d$ be the input, $\mathbf{W}^{(1)} \in \mathbb{R}^{m\times d}$ is the first weight matrix, $\mathbf{W}^{(h)} \in \mathbb{R}^{m\times m}$ is the weight at the h-th layer for $2 \le h \le H$, $\mathbf{a} \in \mathbb{R}^m$ is the output layer and $\sigma(\cdot)$ is the activation function.[3] We define the prediction function recursively (for simplicity we let $\mathbf{x}^{(0)} = \mathbf{x}$).

$$\mathbf{x}^{(h)} = \sqrt{\frac{c_\sigma}{m}}\sigma\left(\mathbf{W}^{(h)}\mathbf{x}^{(h-1)}\right), 1 \le h \le H$$
$$f(\mathbf{x}, \theta) = \mathbf{a}^\top \mathbf{x}^{(H)}. \tag{2}$$

  where $c_\sigma = \left(\mathbb{E}_{x\sim N(0,1)}\left[\sigma(x)^2\right]\right)^{-1}$ is a scaling factor to normalize the input in the initialization phase.

- **ResNet**[4]: We use the same notations as the multilayer fully connected neural networks. We define the prediction recursively.

$$\mathbf{x}^{(1)} = \sqrt{\frac{c_\sigma}{m}}\sigma\left(\mathbf{W}^{(1)}\mathbf{x}\right),$$
$$\mathbf{x}^{(h)} = \mathbf{x}^{(h-1)} + \frac{c_{res}}{H\sqrt{m}}\sigma\left(\mathbf{W}^{(h)}\mathbf{x}^{(h-1)}\right) \quad \text{for } 2 \le h \le H,$$
$$f_{res}(\mathbf{x}, \theta) = \mathbf{a}^\top \mathbf{x}^{(H)} \tag{3}$$

  where $0 < c_{res} < 1$ is a small constant. Note here we use a $\frac{c_{res}}{H\sqrt{m}}$ scaling. This scaling plays an important role in guaranteeing the width per layer only needs to scale polynomially with H. In practice, the small scaling is enforced by a small initialization of the residual connection (Hardt & Ma, 2016; Zhang et al., 2019), which obtains state-of-the-art performance for deep residual networks. We choose to use an explicit scaling, instead of altering the initialization scheme for notational convenience.

- **Convolutional ResNet**: Lastly, we consider the convolutional ResNet architecture. Again we define the prediction function in a recursive way. Let $\mathbf{x}^{(0)} \in \mathbb{R}^{d_0\times p}$ be the input, where d_0 is the number of input channels and p is the number of pixels. For $h \in [H]$, we let the number of channels be $d_h = m$ and number of pixels be p. Given $\mathbf{x}^{(h-1)} \in \mathbb{R}^{d_{h-1}\times p}$ for $h \in [H]$, we first use an operator $\phi_h(\cdot)$ to divide $\mathbf{x}^{(h-1)}$ into p patches. Each patch has size qd_{h-1} and this implies a map $\phi_h(\mathbf{x}^{(h-1)}) \in \mathbb{R}^{qd_{h-1}\times p}$. For example, when the stride is 1 and $q = 3$

$$\phi_h(\mathbf{x}^{(h-1)}) = \begin{pmatrix} \left(\mathbf{x}^{(h-1)}_{1,0:2}\right)^\top, & \cdots & , \left(\mathbf{x}^{(h-1)}_{1,p-1:p+1}\right)^\top \\ \cdots, & \cdots, & \cdots \\ \left(\mathbf{x}^{(h-1)}_{d_{h-1},0:2}\right)^\top, & \cdots, & \left(\mathbf{x}^{(h-1)}_{d_{h-1},p-1:p+1}\right)^\top \end{pmatrix}$$

  where we let $\mathbf{x}^{(h-1)}_{:,0} = \mathbf{x}^{(h-1)}_{:,p+1} = \mathbf{0}$, i.e., zero-padding. Note this operator has the property

$$\left\|\mathbf{x}^{(h-1)}\right\|_F \le \left\|\phi_h(\mathbf{x}^{(h-1)})\right\|_F \le \sqrt{q}\left\|\mathbf{x}^{(h-1)}\right\|_F.$$

  because each element from $\mathbf{x}^{(h-1)}$ at least appears once and at most appears q times. In practice, q is often small like 3×3, so throughout the paper we view q as a constant in our theoretical analysis. To proceed, let $\mathbf{W}^{(h)} \in \mathbb{R}^{d_h\times qd_{h-1}}$, we have

$$\mathbf{x}^{(1)} = \sqrt{\frac{c_\sigma}{m}}\sigma\left(\mathbf{W}^{(1)}\phi_1(\mathbf{x})\right) \in \mathbb{R}^{m\times p},$$
$$\mathbf{x}^{(h)} = \mathbf{x}^{(h-1)} + \frac{c_{res}}{H\sqrt{m}}\sigma\left(\mathbf{W}^{(h)}\phi_h(\mathbf{x}^{(h-1))}\right) \in \mathbb{R}^{m\times p} \quad \text{for } 2 \le h \le H,$$

  where $0 < c_{res} < 1$ is a small constant. Finally, for $\mathbf{a} \in \mathbb{R}^{m\times p}$, the output is defined as

$$f_{cnn}(\mathbf{x}, \theta) = \langle \mathbf{a}, \mathbf{x}^{(H)} \rangle.$$

  Note here we use the similar scaling $O(\frac{1}{H\sqrt{m}})$ as ResNet.

To learn the deep neural network, we consider the randomly initialized gradient descent algorithm to find the global minimizer of the empirical loss (1). Specifically, we use the following random initialization scheme. For every level $h \in [H]$, each entry is sampled from a standard Gaussian distribution, $\mathbf{W}^{(h)}_{ij} \sim N(0, 1)$ and each entry of the output layer $\mathbf{a}$ is also sampled from $N(0, 1)$. In this paper, we train all layers by gradient descent, for $k = 1, 2, \ldots,$ and $h \in [H]$

$$\mathbf{W}^{(h)}(k) = \mathbf{W}^{(h)}(k-1) - \eta\frac{\partial L(\theta(k-1))}{\partial \mathbf{W}^{(h)}(k-1)},$$
$$\mathbf{a}(k) = \mathbf{a}(k-1) - \eta\frac{\partial L(\theta(k-1))}{\partial \mathbf{a}(k-1)}$$

where $\eta > 0$ is the step size.

[3] We assume intermediate layers are square matrices for simplicity. It is not difficult to generalize our analysis to rectangular weight matrices.

[4] We will refer to this architecture as ResNet, although this differs by the standard ResNet architecture since the skip-connections at every layer, instead of every two layers. This architecture was previously studied in (Hardt & Ma, 2016). We study this architecture for the ease of presentation and analysis. It is not hard to generalize our analysis to architectures with skip-connections are every two or more layers.

4. Technique Overview

In this section, we describe our main idea of proving the global convergence of gradient descent. Our proof technique is inspired by Du et al. (2018b) who proposed to study the dynamics of differences between labels and predictions. Here the individual prediction at the k-th iteration is

$$u_i(k) = f(\theta(k), \mathbf{x}_i)$$

and we denote $\mathbf{u}(k) = (u_1(k), \ldots, u_n(k))^\top \in \mathbb{R}^n$. Du et al. (2018b) showed that for two-layer fully-connected neural network, the sequence $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ admits the following dynamics

$$\mathbf{y} - \mathbf{u}(k+1) = (\mathbf{I} - \eta \mathbf{H}(k)) (\mathbf{y} - \mathbf{u}(k))$$

where $\mathbf{H}(k) \in \mathbb{R}^{n \times n}$ is a Gram matrix with[5]

$$\mathbf{H}_{ij}(k) = \left\langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(1)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(1)}(k)} \right\rangle.$$

The key finding in (Du et al., 2018b) is that if m is sufficiently large, $\mathbf{H}(k) \approx \mathbf{H}^\infty$ for all k where $\mathbf{H}^\infty$ is defined as $\mathbf{H}_{ij}^\infty = \mathbb{E}_{\mathbf{w} \sim N(\mathbf{0}, \mathbf{I})} \left[\sigma'\left(\mathbf{w}^\top \mathbf{x}_i\right) \sigma'\left(\mathbf{w}^\top \mathbf{x}_j\right) \mathbf{x}_i^\top \mathbf{x}_j \right]$. Notably, $\mathbf{H}^\infty$ is a fixed matrix which only depends on the training input, but *does not* depend on neural network parameters θ. As a direct result, in the large m regime, the dynamics of $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ is approximately *linear*

$$\mathbf{y} - \mathbf{u}(k+1) \approx (\mathbf{I} - \eta \mathbf{H}^\infty) (\mathbf{y} - \mathbf{u}(k)).$$

For this linear dynamics, using standard analysis technique for power method, one can show $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ converges to $\mathbf{0}$ where the rate is determined by the least eigenvalue of $\mathbf{H}^\infty$ and the step size η.

We leverage this insight to our deep neural network setting. Again we consider the sequence $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$, which admits the dynamics

$$\mathbf{y} - \mathbf{u}(k+1) = (\mathbf{I} - \eta \mathbf{G}(k)) (\mathbf{y} - \mathbf{u}(k))$$

where

$$\begin{aligned}
&\mathbf{G}_{ij}(k) \\
&= \left\langle \frac{\partial u_i(k)}{\partial \theta(k)}, \frac{\partial u_j(k)}{\partial \theta(k)} \right\rangle \\
&= \sum_{h=1}^{H} \left\langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(h)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(h)}(k)} \right\rangle + \left\langle \frac{\partial u_i(k)}{\partial \mathbf{a}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{a}(k)} \right\rangle \\
&\triangleq \sum_{h=1}^{H+1} \mathbf{G}_{ij}^{(h)}(k).
\end{aligned}$$

[5]This formula is for the setting that only the first layer is trained.

Here we define $\mathbf{G}^{(h)} \in \mathbb{R}^{n \times n}$ with $\mathbf{G}_{ij}^{(h)}(k) = \left\langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(h)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(h)}(k)} \right\rangle$ for $h = 1, \ldots, H$ and $\mathbf{G}_{ij}^{(H+1)}(k) = \left\langle \frac{\partial u_i(k)}{\partial \mathbf{a}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{a}(k)} \right\rangle$. Note for all $h \in [H+1]$, each entry of $\mathbf{G}^{(h)}(k)$ is an inner product. Therefore, $\mathbf{G}^{(h)}(k)$ is a positive semi-definite (PSD) matrix for $h \in [H+1]$. Furthermore, if there exists one $h \in [H]$ that $\mathbf{G}^{(h)}(k)$ is strictly positive definite, then if one chooses the step size η to be sufficiently small, the loss decreases at the k-th iteration according the analysis of power method. In this paper we focus on $\mathbf{G}^{(H)}(k)$, the gram matrix induced by the weights from H-th layer for simplicity at the cost of a minor degradation in convergence rate.[6]

We use the similar observation in (Du et al., 2018b) that we show if the width is large enough for all layers, for all $k = 0, 1, \ldots$, $\mathbf{G}^{(H)}(k)$ is close to a fixed matrix $\mathbf{K}^{(H)} \in \mathbb{R}^{n \times n}$ which depends on the input data, neural network architecture and the activation but does not depend on neural network parameters θ. According to the analysis of the power method, once we establish this, as long as $\mathbf{K}^{(H)}$ is strictly positive definite, then the gradient descent enjoys a linear convergence rate. We will show for $\mathbf{K}^{(H)}$ is strictly positive definite as long as the training data is not degenerate (c.f. Proposition F.1 and F.2).

While following the similar high-level analysis framework proposed by Du et al. (2018b), analyzing the convergence of gradient descent for *deep* neural network is significantly more involved and requires new technical tools. To show $\mathbf{G}^{(H)}(k)$ is close to $\mathbf{K}^{(H)}$, we have two steps. First, we show in the initialization phase $\mathbf{G}^{(H)}(0)$ is close to $\mathbf{K}^{(H)}$. Second, we show during training $\mathbf{G}^{(H)}(k)$ is close to $\mathbf{G}^{(H)}(0)$ for $k = 1, 2, \ldots$. Below we give overviews of these two steps.

Analysis of Random Initialization Unlike (Du et al., 2018b) in which they showed $\mathbf{H}(0)$ is close to $\mathbf{H}^\infty$ via a simple concentration inequality, showing $\mathbf{G}^{(H)}(0)$ is close to $\mathbf{K}^{(H)}$ requires more subtle calculations. First, as will be clear in the following sections, $\mathbf{K}^{(H)}$ is a recursively defined matrix. Therefore, we need to analyze how the perturbation (due to randomness of initialization and finite m) from lower layers propagates to the H-th layer. Second, this perturbation propagation involves non-linear operations due to the activation function. To quantitatively characterize this perturbation propagation dynamics, we use induction and leverage techniques from Malliavin calculus (Malliavin, 1995). We derive a general framework that allows us to analyze the initialization behavior for the fully-connected neural network, ResNet, convolutional ResNet and other potential neural network architectures in a unified way.

[6]Using the contribution of all the gram matrices to the minimum eigenvalue can potentially improve the convergence rate.

One important finding in our analysis is that ResNet architecture makes the "perturbation propagation" more stable. The high level intuition is the following. For fully connected neural network, suppose we have some perturbation $\left\|\mathbf{G}^{(1)}(0)-\mathbf{K}^{(1)}\right\|_2 \leq \mathcal{E}_1$ in the first layer. This perturbation propagates to the H-th layer admits the form

$$\left\|\mathbf{G}^{(H)}(0)-\mathbf{K}^{(H)}\right\|_2 \triangleq \mathcal{E}_H \lesssim 2^{O(H)}\mathcal{E}_1. \tag{4}$$

Therefore, we need to have $\mathcal{E}_1 \leq \frac{1}{2^{O(H)}}$ and this makes m have exponential dependency on H.[7]

On the other hand, for ResNet the perturbation propagation admits the form

$$\mathcal{E}_H \lesssim \left(1+O\left(\frac{1}{H}\right)\right)^H \epsilon_1 = O\left(\epsilon_1\right) \tag{5}$$

Therefore we do not have the exponential explosion problem for ResNet. We refer readers to Section E for details.

Analysis of Perturbation of During Training The next step is to show $\mathbf{G}^{(H)}(k)$ is close to $\mathbf{G}^{(H)}(0)$ for $k=0,1,\ldots$. Note $\mathbf{G}^{(H)}$ depends on weight matrices from all layers, so to establish that $\mathbf{G}^{(H)}(k)$ is close to $\mathbf{G}^{(H)}(0)$, we need to show $\mathbf{W}^{(h)}(k)-\mathbf{W}^{(h)}(0)$ is small for all $h\in[H]$ and $\mathbf{a}(k)-\mathbf{a}(0)$ is small.

In the two-layer neural network setting (Du et al., 2018b), they are able to show *every* weight vector of the first layer is close to its initialization, i.e., $\left\|\mathbf{W}^{(1)}(k)-\mathbf{W}^{(1)}(0)\right\|_{2,\infty}$ is small for $k=0,1,\ldots$. While establishing this condition for two-layer neural network is not hard, this condition may not hold for multi-layer neural networks. In this paper, we show instead, the averaged Frobenius norm

$$\frac{1}{\sqrt{m}}\left\|\mathbf{W}^{(h)}(k)-\mathbf{W}^{(h)}(0)\right\|_F \tag{6}$$

is small for all $k=0,1,\ldots$.

Similar to the analysis in the initialization, showing Equation (6) is small is highly involved because again, we need to analyze how the perturbation propagates. We develop a unified proof strategy for the fully-connected neural network, ResNet and convolutional ResNet. Our analysis in this step again sheds light on the benefit of using ResNet architecture for training. The high-level intuition is similar to Equation (5). See Section B, C, and D for details.

[7] We not mean to imply that fully-connected networks necessarily depend exponentially on H, but simply to illustrate in our analysis why the exponential dependence arises. For specific activations such as ReLU and careful initialization schemes, this exponential dependence may be avoided.

5. Warm Up: Convergence Result of GD for Deep Fully-connected Neural Networks

In this section, as a warm up, we show gradient descent with a constant positive step size converges to the global minimum at a linear rate. As we discussed in Section 4, the convergence rate depends on least eigenvalue of the Gram matrix $\mathbf{K}^{(H)}$.

Definition 5.1. *The Gram matrix $\mathbf{K}^{(H)}$ is recursively defined as follows, for $(i,j)\in[n]\times[n]$, and $h=1,\ldots,H-1$*

$$\begin{aligned}
\mathbf{K}_{ij}^{(0)} =& \langle \mathbf{x}_i, \mathbf{x}_j\rangle, \\
\mathbf{A}_{ij}^{(h)} =& \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix}, \\
\mathbf{K}_{ij}^{(h)} =& c_\sigma \mathbb{E}_{(u,v)^\top\sim N\left(\mathbf{0},\mathbf{A}_{ij}^{(h)}\right)}\left[\sigma\left(u\right)\sigma\left(v\right)\right], \\
\mathbf{K}_{ij}^{(H)} =& c_\sigma \mathbf{K}_{ij}^{(H-1)} \mathbb{E}_{(u,v)^\top\sim N\left(\mathbf{0},\mathbf{A}_{ij}^{(H-1)}\right)}\left[\sigma'(u)\sigma'(v)\right].
\end{aligned} \tag{7}$$

The derivation of this Gram matrix is deferred to Section E. The convergence rate and the amount of overparameterization depends on the least eigenvalue of this Gram matrix. In Section F.1 we show as long as the input training data is not degenerate, then $\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$ is strictly positive. We remark that if $H=1$, then $\mathbf{K}^{(H)}$ is the same the Gram matrix defined in (Du et al., 2018b).

Now we are ready to state our main convergence result of gradient descent for deep fully-connected neural networks.

Theorem 5.1 (Convergence Rate of Gradient Descent for Deep Fully-connected Neural Networks)**.** *Assume for all $i\in[n]$, $\|\mathbf{x}_i\|_2=1$, $|y_i|=O(1)$ and the number of hidden nodes per layer*

$$m=\Omega\left(2^{O(H)}\max\left\{\frac{n^4}{\lambda_{\min}^4\left(\mathbf{K}^{(H)}\right)},\frac{n}{\delta},\frac{n^2\log(\frac{Hn}{\delta})}{\lambda_{\min}^2\left(\mathbf{K}^{(H)}\right)}\right\}\right)$$

where $\mathbf{K}^{(H)}$ is defined in Equation (7). If we set the step size

$$\eta=O\left(\frac{\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{n^2 2^{O(H)}}\right),$$

then with probability at least $1-\delta$ over the random initialization the loss, for $k=1,2,\ldots$, the loss at each iteration satisfies

$$L(\theta(k))\leq\left(1-\frac{\eta\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{2}\right)^k L(\theta(0)).$$

This theorem states that if the width m is large enough and we set step size appropriately then gradient descent converges to the global minimum with zero loss at linear

rate. The main assumption of the theorem is that we need a large enough width of each layer. The width m depends on n, H and $1/\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$. The dependency on n is only polynomial, which is the same as previous work on shallow neural networks (Du et al., 2018b; Li & Liang, 2018). Similar to (Du et al., 2018b), m also polynomially depends on $1/\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$. However, the dependency on the number of layers H is exponential. As we discussed in Section B.1, this exponential comes from the instability of the fully-connected architecture (c.f. Equation (4)). In the next section, we show with ResNet architecture, we can reduce the dependency on H from $2^{(H)}$ to $\mathrm{poly}(H)$.

Note the requirement of m has three terms. The first term is used to show the Gram matrix is stable during training. The second term is used to guarantee the output in each layer is approximately normalized at the initialization phase. The third term is used to show the perturbation of Gram matrix at the initialization phase is small. See Section B for proofs.

The convergence rate depends step size η and $\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$, similar to (Du et al., 2018b). Here we require $\eta = O\left(\frac{\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{n^2 2^{O(H)}}\right)$. When $H = 1$, this requirement is the same as the one used in (Du et al., 2018b). However, for deep fully-connected neural network, we require η to be exponentially small in terms of number of layers. The reason is similar to that we require m to be exponentially large. Again, this will be improved in the next section.

6. Convergence Result of GD for ResNet

In this section we consider the convergence of gradient descent for training a ResNet. We will focus on how much over-parameterization is needed to ensure the global convergence of gradient descent and compare it with fully-connected neural networks. Again we first define the key Gram matrix whose least eigenvalue will determine the convergence rate.

Definition 6.1. *The Gram matrix $\mathbf{K}^{(H)}$ is recursively defined as follows, for $(i,j) \in [n]\times[n]$ and $h = 2,\ldots,H-1$:*

$$
\begin{aligned}
\mathbf{K}_{ij}^{(0)} =& \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\
\mathbf{K}_{ij}^{(1)} =& \mathbb{E}_{(u,v)^\top \sim N\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_{ii}^{(0)} & \mathbf{K}_{ij}^{(0)} \\ \mathbf{K}_{ji}^{(0)} & \mathbf{K}_{jj}^{(0)} \end{pmatrix}\right)} c_\sigma \sigma(u)\sigma(v), \\
\mathbf{b}_i^{(1)} =& \sqrt{c_\sigma}\mathbb{E}_{u\sim N\left(0,\mathbf{K}_{ii}^{(0)}\right)}\left[\sigma(u)\right], \\
\mathbf{A}_{ij}^{(h)} =& \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix} \\
\mathbf{K}_{ij}^{(h)} =& \mathbf{K}_{ij}^{(h-1)} + \\
& \mathbb{E}_{(u,v)^\top \sim N\left(\mathbf{0}, \mathbf{A}_{ij}^{(h)}\right)} \left[\frac{c_{res}\mathbf{b}_i^{(h-1)}\sigma(u)}{H} + \frac{c_{res}\mathbf{b}_j^{(h-1)}\sigma(v)}{H} + \frac{c_{res}^2\sigma(u)\sigma(v)}{H^2} \right], \\
\mathbf{b}_i^{(h)} =& \mathbf{b}_i^{(h-1)} + \frac{c_{res}}{H}\mathbb{E}_{u\sim N\left(0,\mathbf{K}_{ii}^{(h-1)}\right)}\left[\sigma(u)\right], \\
\mathbf{K}_{ij}^{(H)} =& \frac{c_{res}^2}{H^2}\mathbf{K}_{ij}^{(H-1)}\mathbb{E}_{(u,v)^\top \sim N\left(\mathbf{0},\mathbf{A}_{ij}^{(H-1)}\right)}\left[\sigma'(u)\sigma'(v)\right].
\end{aligned} \tag{8}
$$

Comparing $\mathbf{K}^{(H)}$ of the ResNet and the one of the fully-connect neural network, the definition of $\mathbf{K}^{(H)}$ also depends on a series of $\{\mathbf{b}^{(h)}\}_{h=1}^{H-1}$. This dependency is comes from the skip connection block in the ResNet architecture. See Section E. In Section F.2, we show as long as the input training data is not degenerate, then $\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$ is strictly positive. Furthermore, $\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$ does not depend inversely exponentially in H.

Now we are ready to state our main theorem for ResNet.

Theorem 6.1 (Convergence Rate of Gradient Descent for ResNet)**.** *Assume for all $i \in [n]$, $\|\mathbf{x}_i\|_2 = 1$, $|y_i| = O(1)$ and the number of hidden nodes per layer*

$$
\begin{aligned}
m =& \Omega\left(\max\left\{\frac{n^4}{\lambda_{\min}^4\left(\mathbf{K}^{(H)}\right)H^6}, \frac{n^2}{\lambda_{\min}^2(\mathbf{K}^{(H)})H^2}, \right.\right. \\
& \left.\left. \frac{n}{\delta}, \frac{n^2\log\left(\frac{Hn}{\delta}\right)}{\lambda_{\min}^2\left(\mathbf{K}^{(H)}\right)}\right\}\right).
\end{aligned} \tag{9}
$$

If we set the step size $\eta = O\left(\frac{\lambda_{\min}\left(\mathbf{K}^{(H)}\right)H^2}{n^2}\right)$, then with probability at least $1-\delta$ over the random initialization we have for $k = 1, 2, \ldots$

$$
L(\theta(k)) \leq \left(1 - \frac{\eta\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{2}\right)^k L(\theta(0)).
$$

In sharp contrast to Theorem 5.1, this theorem is fully polynomial in the sense that both the number of neurons and the convergence rate is polynomially in n and H. Note the amount of over-parameterization depends on $\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$ which is the smallest eigenvalue of the H-th layer's Gram matrix. The main reason that we do not have any exponential factor here is that the skip connection block makes the overall architecture more stable in both the initialization phase and the training phase.

Note the requirement on m has 4 terms. The first two terms are used to show the Gram matrix stable during training. The third term is used to guarantee the output in each layer is approximately normalized at the initialization phase. The fourth term is used to show bound the size of the perturbation of the Gram matrix at the initialization phase. See Section C for details.

7. Convergence Result of GD for Convolutional ResNet

In this section we generalize the convergence result of gradient descent for ResNet to convolutional ResNet. Again, we focus on how much over-parameterization is needed to ensure the global convergence of gradient descent. Similar to previous sections, we first define the $\mathbf{K}^{(H)}$ for this architecture.

Definition 7.1. *The Gram matrix $\mathbf{K}^{(H)}$ is recursively defined as follows, for $(i,j) \in [n] \times [n]$, $(l,r) \in [p] \times [p]$ and $h = 2, \ldots, H-1$,*

$$
\begin{aligned}
\mathbf{K}_{ij}^{(0)} =& \phi_1(\mathbf{x}_i)^\top \phi_1(\mathbf{x}_j) \in \mathbb{R}^{p\times p}, \\
\mathbf{K}_{ij}^{(1)} =& \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim N\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_{ii}^{(0)} & \mathbf{K}_{ij}^{(0)} \\ \mathbf{K}_{ji}^{(0)} & \mathbf{K}_{jj}^{(0)} \end{pmatrix}\right)} c_\sigma \sigma(\mathbf{u})^\top \sigma(\mathbf{v}), \\
\mathbf{b}_i^{(1)} =& \sqrt{c_\sigma}\mathbb{E}_{\mathbf{u}\sim N\left(\mathbf{0}, \mathbf{K}_{ii}^{(0)}\right)} [\sigma(\mathbf{u})], \\
\mathbf{A}_{ij}^{(h)} =& \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix} \\
\mathbf{H}_{ij}^{(h)} =& \mathbf{K}_{ij}^{(h-1)} + \\
& \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim N\left(\mathbf{0}, \mathbf{A}_{ij}^{(h-1)}\right)} \left[\frac{c_{res}\mathbf{b}_i^{(h-1)\top}\sigma(\mathbf{u})}{H} \right. \qquad (10) \\
& \left. + \frac{c_{res}\mathbf{b}_j^{(h-1)\top}\sigma(\mathbf{v})}{H} + \frac{c_{res}^2 \sigma(\mathbf{u})^\top \sigma(\mathbf{v})}{H^2} \right], \\
\mathbf{K}_{ij,lr}^{(h)} =& \mathrm{tr}\left(\mathbf{H}_{ij,D_l^{(h)}D_r^{(h)}}^{(h)}\right), \\
\mathbf{b}_i^{(h)} =& \mathbf{b}_i^{(h-1)} + \frac{c_{res}}{H}\mathbb{E}_{\mathbf{u}\sim N\left(\mathbf{0}, \mathbf{K}_{ii}^{(h-1)}\right)} [\sigma(\mathbf{u})] \\
\mathbf{M}_{ij,lr}^{(H)} =& \mathbf{K}_{ij,lr}^{(H-1)} \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim N\left(\mathbf{0}, \mathbf{A}_{ij}^{(H-1)}\right)} [\sigma'(u_l)\sigma'(v_r)] \\
\mathbf{K}_{ij}^{(H)} =& \mathrm{tr}(\mathbf{M}_{ij}^{(H)})
\end{aligned}
$$

where $\mathbf{u}$ and $\mathbf{v}$ are both random row vectors and $D_l^{(h)} \triangleq \{s : \mathbf{x}_{:,s}^{(h-1)} \in$ the l^{th} patch$\}$.

Note here $\mathbf{K}_{ij}^{(h)}$ has dimension $p \times p$ for $h = 0, \ldots, H-1$ and $\mathbf{K}_{ij,lr}$ denotes the (l,r)-th entry.

Now we state our main convergence theorem for the convolutional ResNet.

Theorem 7.1 (Convergence Rate of Gradient Descent for Convolutional ResNet)**.** *Assume for all $i \in [n]$, $\|\mathbf{x}_i\|_F = 1$, $|y_i| = O(1)$ and the number of hidden nodes per layer*

$$
\begin{aligned}
m =& \Omega\left(\max\left\{\frac{n^4}{\lambda_0^4 H^6}, \frac{n^4}{\lambda_0^4 H^2}, \right.\right. \\
& \left.\left. \frac{n}{\delta}, \frac{n^2 \log\left(\frac{Hn}{\delta}\right)}{\lambda_0^2}\right\} \mathrm{poly}(p)\right). \qquad (11)
\end{aligned}
$$

If we set the step size $\eta = O\left(\frac{\lambda_0 H^2}{n^2 \mathrm{poly}(p)}\right)$, then with probability at least $1-\delta$ over the random initialization we have for $k = 1, 2, \ldots$

$$
L(\theta(k)) \leq \left(1 - \frac{\eta\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{2}\right)^k L(\theta(0)).
$$

This theorem is similar to that of ResNet. The number of neurons required per layer is only polynomial in the depth and the number of data points and step size is only polynomially small. The only extra term is $\mathrm{poly}(p)$ in the requirement of m and η. The analysis is also similar to ResNet and we refer readers to Section D for details.

8. Conclusion

In this paper, we show that gradient descent on deep overparametrized networks can obtain zero training loss. Our proof builds on a careful analysis of the random initialization scheme and a perturbation analysis which shows that the Gram matrix is increasingly stable under overparametrization. These techniques allow us to show that every step of gradient descent decreases the loss at a geometric rate.

We list some directions for future research:

1. The current paper focuses on the training loss, but does not address the test loss. It would be an important problem to show that gradient descent can also find solutions of low test loss. In particular, existing work only demonstrate that gradient descent works under the same situations as kernel methods and random feature methods (Daniely, 2017; Li & Liang, 2018; Allen-Zhu et al., 2018a; Arora et al., 2019). To further investigate of generalization behavior, we believe some algorithm-dependent analyses may be useful (Hardt et al., 2016; Mou et al., 2018; Chen et al., 2018).

2. The width of the layers m is polynomial in all the parameters for the ResNet architecture, but still very large. Realistic networks have number of parameters, not width, a large constant multiple of n. We consider improving the analysis to cover commonly utilized networks an important open problem.

3. The current analysis is for gradient descent, instead of stochastic gradient descent. We believe the analysis can be extended to stochastic gradient, while maintaining the linear convergence rate.

4. The convergence rate can be potentially improved if the minimum eigenvalue takes into account the contribution of all Gram matrices, but this would considerably complicate the initialization and perturbation analysis.

Acknowledgments

We thank Lijie Chen and Ruosong Wang for useful discussions. SSD acknowledges support from AFRL grant FA8750-17-2-0212 and DARPA D17AP00001. JDL acknowledges support of the ARO under MURI Award W911NF-11-1-0303. This is part of the collaboration between US DOD, UK MOD and UK Engineering and Physical Research Council (EPSRC) under the Multidisciplinary University Research Initiative. HL and LW acknowlege support from National Basic Research Program of China (973 Program) (grant no. 2015CB352502), NSFC (61573026) and BJNSF (L172037). Part of the work is done while SSD was visiting Simons Institute.

References

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*, 2018a.

Allen-Zhu, Z., Li, Y., and Song, Z. On the convergence rate of training recurrent neural networks. *arXiv preprint arXiv:1810.12065*, 2018b.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018c.

Andoni, A., Panigrahy, R., Valiant, G., and Zhang, L. Learning polynomials with neural networks. In *International Conference on Machine Learning*, pp. 1908–1916, 2014.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

Brutzkus, A. and Globerson, A. Globally optimal gradient descent for a ConvNet with gaussian inputs. In *International Conference on Machine Learning*, pp. 605–614, 2017.

Chen, Y., Jin, C., and Yu, B. Stability and Convergence Trade-off of Iterative Optimization Algorithms. *arXiv e-prints*, art. arXiv:1804.01619, Apr 2018.

Chizat, L. and Bach, F. On the global convergence of gradient descent for over-parameterized models using optimal transport. *arXiv preprint arXiv:1805.09545*, 2018a.

Chizat, L. and Bach, F. A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018b.

Daniely, A. SGD learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2017.

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

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Singh, A., and Poczos, B. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pp. 1067–1077, 2017a.

Du, S. S., Lee, J. D., and Tian, Y. When is a convolutional filter easy to learn? *arXiv preprint arXiv:1709.06129*, 2017b.

Du, S. S., Lee, J. D., Tian, Y., Poczos, B., and Singh, A. Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima. *Proceedings of the 35th International Conference on Machine Learning*, pp. 1339–1348, 2018a.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018b.

Freeman, C. D. and Bruna, J. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle points − online stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory*, pp. 797–842, 2015.

Haeffele, B. D. and Vidal, R. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*, 2015.

Hardt, M. and Ma, T. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1225–1234, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/hardt16.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1724–1732, 2017.

Kawaguchi, K. Deep learning without poor local minima. In *Advances In Neural Information Processing Systems*, pp. 586–594, 2016.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent only converges to minimizers. In *Conference on Learning Theory*, pp. 1246–1257, 2016.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. *arXiv preprint arXiv:1808.01204*, 2018.

Li, Y. and Yuan, Y. Convergence analysis of two-layer neural networks with ReLU activation. In *Advances in Neural Information Processing Systems*, pp. 597–607, 2017.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems 30*, pp. 6231–6239. Curran Associates, Inc., 2017.

Malliavin, P. Gaussian sobolev spaces and stochastic calculus of variations. 1995.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Mei, S., Montanari, A., and Nguyen, P.-M. A mean field view of the landscape of two-layers neural networks. *Proceedings of the National Academy of Sciences*, pp. E7665–E7671, 2018.

Mou, W., Wang, L., Zhai, X., and Zheng, K. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. In Bubeck, S., Perchet, V., and Rigollet, P. (eds.), *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pp. 605–638. PMLR, 06–09 Jul 2018. URL http://proceedings.mlr.press/v75/mou18a.html.

Nguyen, Q. and Hein, M. The loss surface of deep and wide neural networks. In *International Conference on Machine Learning*, pp. 2603–2612, 2017.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

Rotskoff, G. M. and Vanden-Eijnden, E. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

Safran, I. and Shamir, O. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pp. 774–782, 2016.

Safran, I. and Shamir, O. Spurious local minima are common in two-layer ReLU neural networks. *In International Conference on Machine Learning*, pp. 4433–4441, 2018.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Sirignano, J. and Spiliopoulos, K. Mean field analysis of neural networks. *arXiv preprint arXiv:1805.01053*, 2018.

Soltanolkotabi, M. Learning ReLUs via gradient descent. In *Advances in Neural Information Processing Systems*, pp. 2007–2017, 2017.

Soltanolkotabi, M., Javanmard, A., and Lee, J. D. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2018.

Soudry, D. and Carmon, Y. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

Soudry, D. and Hoffer, E. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *arXiv preprint arXiv:1702.05777*, 2017.

Tian, Y. An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis. In *International Conference on Machine Learning*, pp. 3404–3413, 2017.

Venturi, L., Bandeira, A., and Bruna, J. Neural networks with finite intrinsic dimension have no spurious valleys. *arXiv preprint arXiv:1802.06384*, 2018.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Wei, C., Lee, J. D., Liu, Q., and Ma, T. On the margin theory of feedforward neural networks. *arXiv preprint arXiv:1810.05369*, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *NIN*, 8:35–67, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, H., Dauphin, Y. N., and Ma, T. Residual learning without normalization via better initialization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gsz30cKX.

Zhang, X., Yu, Y., Wang, L., and Gu, Q. Learning one-hidden-layer relu networks via gradient descent. *arXiv preprint arXiv:1806.07808*, 2018.

Zhong, K., Song, Z., and Dhillon, I. S. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017a.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. *arXiv preprint arXiv:1706.03175*, 2017b.

Zhou, Y. and Liang, Y. Critical points of neural networks: Analytical forms and landscape properties. *arXiv preprint arXiv:1710.11205*, 2017.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Stochastic gradient descent optimizes over-parameterized deep ReLU networks. *arXiv preprint arXiv:1811.08888*, 2018.