A Convergence Theory for Deep Learning via Over-Parameterization

Zeyuan Allen-Zhu [*1] **Yuanzhi Li** [*2 3] **Zhao Song** [*4 5 6]

Abstract

Deep neural networks (DNNs) have demonstrated dominating performance in many fields; since AlexNet, networks used in practice are going wider and deeper. On the theoretical side, a long line of works have been focusing on why we can train neural networks when there is only one hidden layer. The theory of multi-layer networks remains unsettled. In this work, we prove simple algorithms such as stochastic gradient descent (SGD) can find *global minima* on the training objective of DNNs in *polynomial time*. We only make two assumptions: the inputs do not degenerate and the network is over-parameterized. The latter means the number of hidden neurons is sufficiently large: *polynomial* in L, the number of DNN layers and in n, the number of training samples. As concrete examples, starting from randomly initialized weights, we show that SGD attains 100% training accuracy in classification tasks, or minimizes regression loss in linear convergence speed $\varepsilon \propto e^{-\Omega(T)}$, with running time polynomial in n and L. Our theory applies to the widely-used but non-smooth ReLU activation, and to any smooth and possibly non-convex loss functions. In terms of network architectures, our theory at least applies to fully-connected neural networks, convolutional neural networks (CNN), and residual neural networks (ResNet).

[*]Equal contribution . Full version and future updates are available at `https://arxiv.org/abs/1811.03962`.

This paper is a follow up to the recurrent neural network (RNN) paper (Allen-Zhu et al., 2018b) by the same set of authors. Most of the techniques used in this paper were already discovered in the RNN paper, and this paper can be viewed as a simplification (or to some extent a special case) of the RNN setting in order to reach out to a wider audience. We compare the difference and mention our additional contribution in Section 1.2.

[1]Microsoft Research AI [2]Stanford University [3]Princeton University [4]UT-Austin [5]University Washington [6]Harvard University. Correspondence to: Zeyuan Allen-Zhu <zeyuan@csail.mit.edu>, Yuanzhi Li <yuanzhil@stanford.edu>, Zhao Song <zhaos@utexas.edu>.

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

1 Introduction

Neural networks have demonstrated a great success in numerous machine-learning tasks (Amodei et al., 2016; Graves et al., 2013; He et al., 2016; Krizhevsky et al., 2012; Lillicrap et al., 2015; Silver et al., 2016; 2017). One of the empirical findings is that neural networks, trained by first-order methods from random initialization, have a remarkable ability of fitting training data (Zhang et al., 2017).

From a capacity perspective, the ability to fit training data may not be surprising: modern neural networks are always heavily over-parameterized — they have (much) more parameters than the total number of training samples. Thus, there exists parameter choices to achieve zero training error as long as data does not degenerate.

Yet, from an optimization perspective, the fact that randomly initialized first-order methods can find optimal solutions on the training data is *quite non-trivial*: neural networks are often equipped with the ReLU activation, making the training objective not only non-convex, but even non-smooth. Even the general convergence for finding approximate critical points of a non-convex, non-smooth function is not fully understood (Burke et al., 2005), and appears to be a challenging question on its own. This is in direct contrast to practice, in which ReLU networks trained by stochastic gradient descent (SGD) from random initialization *almost never* face the problem of non-smoothness or non-convexity, and can converge to even a *global minimal* over the training set quite easily. This was demonstrated by Goodfellow et al. (2015) using experiments for a variety of network architectures, and a theoretical justification *remains missing* to explain this phenomenon.

Recently, there are quite a few papers trying to understand the success of neural networks from optimization perspective. Many of them focus on the case when the inputs are random Gaussian, and work only for two-layer neural networks (Brutzkus & Globerson, 2017; Du et al., 2018b; Ge et al., 2017; Li & Yuan, 2017; Panigrahy et al., 2018; Soltanolkotabi, 2017; Tian, 2017; Zhong et al., 2017a;b).

In Li & Liang (2018), it was shown that for a two-layer network with ReLU activation, SGD finds nearly-global optimal (say, 99% classification accuracy) solutions on the training data, as long as the network is *over-parameterized*, meaning that when the number of neurons is polynomi-

ally large comparing to the input size. Moreover, if the data is sufficiently structured (say, coming from mixtures of separable distributions), this perfect accuracy extends also to *test data*. As a separate note, over-parameterization is suggested as the possible key to avoid bad local minima by Safran & Shamir (2018) even for two-layer networks.

There are also results that go beyond two-layer neural networks but with limitations. Some consider deep *linear* neural networks without any activation functions (Arora et al., 2018a; Bartlett et al., 2018; Hardt & Ma, 2017; Kawaguchi, 2016). The result of Daniely (2017) applies to multi-layer neural network with ReLU activation, but is about the *convex* training process only with respect to the last layer. Daniely worked in a parameter regime where the weight changes of all layers except the last one make negligible contribution to the final output (and they form the so-called conjugate kernel). The result of Soudry & Carmon (2016) shows that under over-parameterization and under random input perturbation, there is bad local minima for multi-layer neural networks. Their work did not show any provable convergence rate.

In this paper, we study the following fundamental question

Can DNN be trained close to zero training error efficiently under mild assumptions?

If so, can the running time depend only polynomially in the number of layers?

Motivation. In 2012 AlexNet (Krizhevsky et al., 2012) was born with 5 convolutional layers. Since then, the common trend in the deep learning community is to build network architectures that go deeper. In 2014, Simonyan & Zisserman (2014) proposed a VGG network with 19 layers. Later, Szegedy et al. (2015) proposed GoogleNet with 22 layers. In practice, we cannot make the network deeper by naively stacking layers together due to the so-called vanishing / exploding gradient issues. For this reason, in 2015, He et al. (2016) proposed an ingenious deep network structure called Deep Residual Network (ResNet), with the capability of handling at least 152 layers. For more overview and variants of ResNet, we refer the readers to (Fung, 2017).

Compared to the practical neural networks that go much deeper, the existing theory has been mostly around two-layer (thus one-hidden-layer) networks even just for the training process alone. It is natural to ask if we can theoretically understand how the training process has worked for multi-layer neural networks.

1.1 Our Result

In this paper, we extend the over-parameterization theory to *multi-layer* neural networks. We show that over-parameterized neural networks can indeed be trained by regular first-order methods to global minima (e.g. *zero* training error), as as long as the dataset is non-degenerate. We say that the dataset is non-degenerate if the data points are distinct. This is a minimal requirement since a dataset $\{(x_1, y_1), (x_2, y_2)\}$ with the same input $x_1 = x_2$ and different labels $y_1 \neq y_2$ can not be trained to zero error. We denote by δ the minimum (relative) distance between two training data points, and by n the number of samples in the training dataset.

Now, consider an L-layer fully-connected feedforward neural network, each layer consisting of m neurons equipped with ReLU activation. We show that,

- As long as $m \geq \mathsf{poly}(n, L, \delta^{-1})$, starting from random Gaussian initialized weights, gradient descent (GD) and stochastic gradient descent (SGD) find ε-error global minimum in ℓ_2 regression using at most $T = \mathsf{poly}(n, L, \delta^{-1}) \log \frac{1}{\varepsilon}$ iterations. This is a linear convergence rate.
- Using the same network, if the task is multi-label classification, then GD and SGD find an 100% accuracy classifier on the training set in $T = \mathsf{poly}(n, L, \delta^{-1})$ iterations.
- Our result also applies to other Lipschitz-smooth loss functions, and some other network architectures including convolutional neural networks (CNNs) and residual networks (ResNet).

Remark. This paper does not cover the the generalization of over-parameterized neural networks to the test data. We refer interested readers to some practical evidence (Srivastava et al., 2015; Zagoruyko & Komodakis, 2016) that deeper (and wider) neural networks actually generalize better. As for theoretical results, over-parameterized neural networks provably generalize at least for two-layer networks (Allen-Zhu et al., 2018a; Li & Liang, 2018) and for three-layer networks (Allen-Zhu et al., 2018a).[1]

A concurrent but different result. We acknowledge a concurrent work of Du et al. (2018a) which has a similar abstract to this paper, but is different from us in many aspects. Since we noticed many readers cannot tell the two results apart, we compare them carefully below. Du et al. (2018a) has two main results, one for fully-connected networks and the other for residual networks (ResNet).

For fully-connected networks, they only proved the training time is no more than *exponential* in the number of layers, leading to a claim of the form "ResNet has an advantage because ResNet is polynomial-time but fully-connected net-

[1] If data is "well-structured" two-layer over-parameterized neural networks can learn it using SGD with polynomially many samples (Li & Liang, 2018). If data is produced by some unknown two-layer (resp. three-layer) neural network, then two-layer (resp. three-layer) neural networks can also provably learn it using SGD and polynomially many samples (Allen-Zhu et al., 2018a).

work is (possibly) exponential-time." As we prove in this paper, fully-connected networks do have *polynomial* training time, so their logic behind this claim is ungrounded.

For residual networks, their training time scales polynomial in $\frac{1}{\lambda_0}$, a parameter that depends on the minimal singular value of a complicated, L-times recursively-defined kernel matrix. It is not clear whether $\frac{1}{\lambda_0}$ is small or even polynomial from their original writing. In their version 2, they have sketched a possible proof to bound $\frac{1}{\lambda_0}$ in the special case of residual networks.

Their result is different from us in many other aspects. Their result only applies to the (significantly simpler[2]) smooth activation functions and thus *cannot* apply to the state-of-the-art ReLU activation. Their ResNet requires the value of weight initialization to be a function polynomial in λ (which is our δ); this can heavily depend on the input data. Their result only applies to gradient descent but not to SGD. Their result only applies to ℓ_2 loss but not others.

1.2 Other Related Works

Li & Liang (2018) originally proved their result for the cross-entropy loss. Later, the "training accuracy" (not the testing accuracy) part of (Li & Liang, 2018) was extended to the ℓ_2 loss (Du et al., 2018c).

Linear networks without activation functions are important subjects on its own. Besides the already cited references (Arora et al., 2018a; Bartlett et al., 2018; Hardt & Ma, 2017; Kawaguchi, 2016), there are a number of works that study *linear dynamical systems*, which can be viewed as the linear version of recurrent neural networks or reinforcement learning. Recent works in this line of research include (Alaeddini et al., 2018; Arora et al., 2018b; Dean et al., 2017; 2018; Hardt et al., 2018; Hazan et al., 2017; 2018; Marecek & Tchrakian, 2018; Oymak & Ozay, 2018; Simchowitz et al., 2018).

There is sequence of work about one-hidden-layer (multiple neurons) CNN (Brutzkus & Globerson, 2017; Du et al., 2018b; Goel et al., 2018; Oymak, 2018; Zhong et al., 2017a). Whether the patches overlap or not plays a crucial role in analyzing algorithms for such CNN. One category of the results have required the patches to be disjoint (Brutzkus & Globerson, 2017; Du et al., 2018b; Zhong et al., 2017a). The other category (Goel et al., 2018; Oymak, 2018) have figured out a weaker assumption or even removed that patch-disjoint assumption. On input data distribution, most relied on inputs being Gaussian (Brutzkus & Globerson, 2017; Du et al., 2018b; Oymak, 2018; Zhong et al., 2017a), and some assumed inputs to be symmetrically distributed with identity covariance and boundedness (Goel et al., 2018).

As for ResNet, Li & Yuan (2017) proved that SGD learns one-hidden-layer residual neural networks under Gaussian input assumption. The techniques in (Zhong et al., 2017a;b) can also be generalized to one-hidden-layer ResNet under the Gaussian input assumption; they can show that GD starting from good initialization point (via tensor initialization) learns ResNet. Hardt & Ma (2017) deep *linear* residual networks have no spurious local optima.

If no assumption is allowed, neural networks have been shown hard in several different perspectives. Thirty years ago, Blum & Rivest (1993) first proved that learning the neural network is NP-complete. Stronger hardness results have been proved over the last decade (Daniely, 2016; Daniely & Shalev-Shwartz, 2016; Goel et al., 2017; Klivans & Sherstov, 2009; Livni et al., 2014; Manurangsi & Reichman, 2018; Song et al., 2017).

An over-parameterized RNN theory. For experts in DNN theory, one may view this present paper as a deeply-simplified version of the recurrent neural network (RNN) paper (Allen-Zhu et al., 2018b) by the same set of authors. A recurrent neural network executed on input sequences with time horizon L is *very similar* to a feedforward neural network with L layers. The main difference is that in feedforward neural networks, weight matrices are different across layers, and thus independently randomly initialized; in contrast, in RNN, the same weight matrix is applied across the entire time horizon so we do not have fresh new randomness for proofs that involve in induction.

So, the over-parameterized convergence theory of DNN is *much simpler* than that of RNN.

We write this DNN result as a separate paper because: (1) not all the readers can easily notice that DNN is easier to study than RNN; (2) we believe the convergence of DNN is important on its own; (3) the proof in this paper is much simpler (30 vs 80 pages) and could reach out to a wider audience; (4) the simplicity of this paper allows us to tighten parameters in some non-trivial ways; and (5) the simplicity of this paper allows us to also study convolutional networks, residual networks, as well as different loss functions (all of them were missing from (Allen-Zhu et al., 2018b)).

We also note that the techniques of this paper can be combined with (Allen-Zhu et al., 2018b) to show the convergence of over-parameterized *deep* RNN.

2 Preliminaries

We use $\mathcal{N}(\mu, \sigma)$ to denote the Gaussian distribution of mean μ and variance σ; and $\mathcal{B}(m, \frac{1}{2})$ to denote the binomial distribution with m trials and $1/2$ success rate. We use $\|v\|$

[2]For instance, we have to establish a semi-smoothness theorem for deep ReLU networks (see Theorem 4). If instead the activation function is Lipscthiz smooth, and if one does not care about exponential blow up in the number of layers L, then the network is automatically $2^{O(L)}$-Lipschitz smooth.

to denote Euclidean norms of vectors v, and $\|\mathbf{M}\|_2, \|\mathbf{M}\|_F$ to denote spectral and Frobenius norms of matrices $\mathbf{M}$. For a tuple $\overrightarrow{\mathbf{W}} = (\mathbf{W}_1, \dots, \mathbf{W}_L)$ of matrices, we let $\|\overrightarrow{\mathbf{W}}\|_2 = \max_{\ell\in[L]} \|\mathbf{W}_\ell\|_2$ and $\|\overrightarrow{\mathbf{W}}\|_F = (\sum_{\ell=1}^L \|\mathbf{W}_\ell\|_F^2)^{1/2}$.

We use $\phi(x) = \max\{0, x\}$ to denote the ReLU function, and extend it to vectors $v \in \mathbb{R}^m$ by letting $\phi(v) = (\phi(v_1), \dots, \phi(v_m))$. We use $\mathbf{1}_{event}$ to denote the indicator function for $event$.

The training data consist of vector pairs $\{(x_i, y_i^*)\}_{i\in[n]}$, where each $x_i \in \mathbb{R}^{\mathfrak{d}}$ is the feature vector and y_i^* is the label of the i-th training sample. We assume *without loss of generality* that data are normalized so that $\|x_i\| = 1$ and its last coordinate $(x_i)_{\mathfrak{d}} = \frac{1}{\sqrt{2}}$.[3] We make the following separable assumption on the training data (motivated by (Li & Liang, 2018)):

Assumption 2.1. *For every pair $i, j \in [n]$, we have $\|x_i - x_j\| \geq \delta$.*

To present the simplest possible proof, the main body of this paper only focuses on depth-L feedforward fully-connected neural networks with an ℓ_2-regression task. Therefore, each $y_i^* \in \mathbb{R}^d$ is a target vector for the regression task. We explain how to extend it to more general settings in Section 5 and the Appendix. For notational simplicity, we assume all the hidden layers have the same number of neurons, and our results trivially generalize to each layer having different number of neurons. Specifically, we focus on the following network

$$\begin{aligned} g_{i,0} &= \mathbf{A}x_i & h_{i,0} &= \phi(\mathbf{A}x_i) & &\text{for } i \in [n] \\ g_{i,\ell} &= \mathbf{W}_\ell h_{i,\ell-1} & h_{i,\ell} &= \phi(\mathbf{W}_\ell h_{i,\ell-1}) & &\text{for } i \in [n], \ell \in [L] \\ y_i &= \mathbf{B}h_{i,L} & & & &\text{for } i \in [n] \end{aligned}$$

where $\mathbf{A} \in \mathbb{R}^{m\times\mathfrak{d}}$ is the weight matrix for the input layer, $\mathbf{W}_\ell \in \mathbb{R}^{m\times m}$ is the weight matrix for the ℓ-th hidden layer, and $\mathbf{B} \in \mathbb{R}^{d\times m}$ is the weight matrix for the output layer. For notational convenience in the proofs, we may also use $h_{i,-1}$ to denote x_i and $\mathbf{W}_0$ to denote $\mathbf{A}$.

Definition 2.2 (diagonal sign matrix)**.** *For each $i \in [n]$ and $\ell \in \{0, 1, \dots, L\}$, we denote by $\mathbf{D}_{i,\ell}$ the diagonal sign matrix where $(\mathbf{D}_{i,\ell})_{k,k} = \mathbf{1}_{(\mathbf{W}_\ell h_{i,\ell-1})_k \geq 0}$ for each $k \in [m]$.*

As a result, we have $h_{i,\ell} = \mathbf{D}_{i,\ell}\mathbf{W}_\ell h_{i,\ell-1} = \mathbf{D}_{i,\ell} g_{i,\ell}$ and $(\mathbf{D}_{i,\ell})_{k,k} = \mathbf{1}_{(g_{i,\ell})_k \geq 0}$.

[3] Without loss of generality, one can re-scale and assume $\|x_i\| \leq 1/\sqrt{2}$ for every $i \in [n]$. Again, without loss of generality, one can pad each x_i by an additional coordinate to ensure $\|x_i\| = 1/\sqrt{2}$. Finally, without loss of generality, one can pad each x_i by an additional coordinate $\frac{1}{\sqrt{2}}$ to ensure $\|x_i\| = 1$. This last coordinate $\frac{1}{\sqrt{2}}$ is equivalent to introducing a (random) bias term, because $\mathbf{A}(\frac{y}{\sqrt{2}}, \frac{1}{\sqrt{2}}) = \frac{\mathbf{A}}{\sqrt{2}}(y, 0) + b$ where $b \sim \mathcal{N}(0, \frac{1}{m}\mathbf{I})$. In our proofs, the specific constant $\frac{1}{\sqrt{2}}$ does not matter.

We make the following standard choices of random initialization:

Definition 2.3. *We say that $\overrightarrow{\mathbf{W}} = (\mathbf{W}_1, \dots, \mathbf{W}_L)$, $\mathbf{A}$ and $\mathbf{B}$ are at random initialization if*

- *$[\mathbf{W}_\ell]_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$ for every $i, j \in [m]$ and $\ell \in [L]$;*
- *$\mathbf{A}_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$ for every $(i, j) \in [m] \times [\mathfrak{d}]$; and*
- *$\mathbf{B}_{i,j} \sim \mathcal{N}(0, \frac{1}{d})$ for every $(i, j) \in [d] \times [m]$.*

Assumption 2.4. *Throughout this paper we assume $m \geq \Omega\big(\mathsf{poly}(n, L, \delta^{-1}) \cdot d\big)$ for some sufficiently large polynomial. To present the simplest proof, we did not try to improve such polynomial factors.*

2.1 Objective and Gradient

Our regression objective is

$$F(\overrightarrow{\mathbf{W}}) := \sum_{i=1}^n F_i(\overrightarrow{\mathbf{W}}) \quad \text{where}$$

$$F_i(\overrightarrow{\mathbf{W}}) := \frac{1}{2}\|\mathbf{B}h_{i,L} - y_i^*\|^2 \quad \text{for each } i \in [n]$$

We also denote by $\mathsf{loss}_i := \mathbf{B}h_{i,L} - y_i^*$ the *loss vector* for sample i. For simplicity, we only focus on training $\overrightarrow{\mathbf{W}}$ in this paper and thus leave $\mathbf{A}$ and $\mathbf{B}$ at random initialization. Our techniques can be extended to the case when $\mathbf{A}$, $\mathbf{B}$ and $\overrightarrow{\mathbf{W}}$ are jointly trained.

Definition 2.5. *For each $\ell \in \{1, 2, \cdots, L\}$, we define $\mathsf{Back}_{i,\ell} := \mathbf{B}\mathbf{D}_{i,L}\mathbf{W}_L \cdots \mathbf{D}_{i,\ell}\mathbf{W}_\ell \in \mathbb{R}^{d\times m}$ and for $\ell = L + 1$, we define $\mathsf{Back}_{i,\ell} = \mathbf{B} \in \mathbb{R}^{d\times m}$.*

Using this notation, one can calculate the gradient of $F(\overrightarrow{\mathbf{W}})$ as follows.

Fact 2.6. *The gradient with respect to the k-th row of $\mathbf{W}_\ell \in \mathbb{R}^{m\times m}$ is*

$$\begin{aligned} &\nabla_{[\mathbf{W}_\ell]_k} F(\overrightarrow{\mathbf{W}}) \\ &= \sum_{i=1}^n (\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)_k \cdot h_{i,\ell-1} \cdot \mathbf{1}_{\langle [\mathbf{W}_\ell]_k, h_{i,\ell-1}\rangle \geq 0} \end{aligned}$$

The gradient with respect to $\mathbf{W}_\ell$ is

$$\nabla_{\mathbf{W}_\ell} F(\overrightarrow{\mathbf{W}}) = \textstyle\sum_{i=1}^n \mathbf{D}_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i) h_{i,\ell-1}^\top$$

We denote by

$$\nabla F(\overrightarrow{\mathbf{W}}) = \big(\nabla_{\mathbf{W}_1} F(\overrightarrow{\mathbf{W}}), \dots, \nabla_{\mathbf{W}_L} F(\overrightarrow{\mathbf{W}})\big) \ .$$

3 Our Results and Techniques

To present our result in the simplest possible way, we choose to mainly focus on fully-connected L-layer neural networks with the ℓ_2 regression loss. We shall extend it to more general settings (such as convolutional and residual networks and other losses) in Section 5. Our main results can be stated as follows:

Theorem 1 (gradient descent)**.** *Suppose $m \geq \widetilde{\Omega}\big(\mathsf{poly}(n, L, \delta^{-1}) \cdot d\big)$. Starting from random initializa-*

tion, with probability at least $1-e^{-\Omega(\log^2 m)}$*, gradient descent with learning rate* $\eta=\Theta\big(\frac{d\delta}{\mathsf{poly}(n,L)\cdot m}\big)$ *finds a point* $F(\overrightarrow{\mathbf{W}})\le\varepsilon$ *in*

$$T=\Theta\big(\frac{\mathsf{poly}(n,L)}{\delta^2}\cdot\log\varepsilon^{-1}\big)$$

iterations.

This is known as the linear convergence rate because ε drops exponentially fast in T. We have not tried to improve the polynomial factors in m and T, and are aware of several ways to improve these factors (but at the expense of complicating the proof). We note that $\mathfrak{d}$ is the data input dimension and our result is independent of $\mathfrak{d}$.

Remark. In our version 1, for simplicity, we also put a $\log^2(1/\varepsilon)$ factor in the amount of over-parameterization m in Theorem 1. Since some readers have raised concerns regarding this Du et al. (2018a), we have now removed it at the expense of changing half a line of the proof.

Theorem 2 (SGD)**.** *Suppose* $b\in[n]$ *and* $m\ge\widetilde{\Omega}\big(\frac{\mathsf{poly}(n,L,\delta^{-1})\cdot d}{b}\big)$*. Starting from random initialization, with probability at least* $1-e^{-\Omega(\log^2 m)}$*, SGD with learning rate* $\eta=\Theta\big(\frac{b\delta d}{\mathsf{poly}(n,L)m\log^2 m}\big)$ *and mini-batch size* b *finds* $F(\overrightarrow{\mathbf{W}})\le\varepsilon$ *in*

$$T=\Theta\big(\frac{\mathsf{poly}(n,L)\cdot\log^2 m}{\delta^2 b}\cdot\log\varepsilon^{-1}\big)$$

iterations.

This is again a linear convergence rate because $T\propto\log\frac{1}{\varepsilon}$. The reason for the additional $\log^2 m$ factor comparing to Theorem 1 is because we have a $1-e^{-\Omega(\log^2 m)}$ high confidence bound.

Remark. For experts in optimization theory, one may immediately question the accuracy of Theorem 2, because SGD is known to converge at a slower rate $T\propto\frac{1}{\mathsf{poly}(\varepsilon)}$ even for convex functions. There is no contradiction here. Imaging a strongly convex function $f(x)=\sum_{i=1}^n f_i(x)$ that has a common minimizer $x^*\in\arg\min_x\{f_i(x)\}$ for every $i\in[n]$, then SGD is known to converge in a linear convergence rate.

3.1 Technical Theorems

The main difficulty of this paper is to prove the following two technical theorems. The first one is about the gradient bounds for points that are sufficiently close to the random initialization:

Theorem 3 (no critical point)**.** *With probability* $\ge 1-e^{-\Omega(m/\mathsf{poly}(n,L,\delta^{-1}))}$ *over randomness* $\overrightarrow{\mathbf{W}}^{(0)},\mathbf{A},\mathbf{B}$*, it satisfies for every* $\ell\in[L]$*, every* $i\in[n]$*, and every* $\overrightarrow{\mathbf{W}}$ *with* $\|\overrightarrow{\mathbf{W}}-\overrightarrow{\mathbf{W}}^{(0)}\|_2\le\frac{1}{\mathsf{poly}(n,L,\delta^{-1})}$*,*

$$\|\nabla F(\overrightarrow{\mathbf{W}})\|_F^2\le O\Big(F(\overrightarrow{\mathbf{W}})\times\frac{Lnm}{d}\Big)$$

$$\|\nabla F(\overrightarrow{\mathbf{W}})\|_F^2\ge\Omega\Big(F(\overrightarrow{\mathbf{W}})\times\frac{\delta m}{dn^2}\Big)\ .$$

Most notably, the second property of Theorem 3 says that as long as the objective is large, the gradient norm is also large. (See also Figure 1.) This means, when we are sufficiently close to the random initialization, there is no saddle point or critical point of any order. This gives us hope to find *global minima* of the objective $F(\overrightarrow{\mathbf{W}})$.

Unfortunately, Theorem 3 itself is enough. Even if we follow the negative gradient direction of $F(\overrightarrow{\mathbf{W}})$, how can we guarantee that the objective truly decreases? Classically in optimization theory, one relies on the smoothness property (e.g. Lipscthiz smoothness (Nesterov, 2004)) to derive such objective-decrease guarantee. Unfortunately, smoothness property at least requires the objective to be *twice differentiable*, but ReLU activation is not.

To deal with this issue, we prove the following "semi-smoothness" property of the objective.

Theorem 4 (semi-smoothness)**.** *With probability at least* $1-e^{-\Omega(m/\mathsf{poly}(L,\log m))}$ *over the randomness of* $\overrightarrow{\mathbf{W}}^{(0)},\mathbf{A},\mathbf{B}$*, we have :*
for every $\overrightarrow{\mathbf{W}}\in(\mathbb{R}^{m\times m})^L$ *with*

$$\|\overrightarrow{\mathbf{W}}-\overrightarrow{\mathbf{W}}^{(0)}\|_2\le\frac{1}{\mathsf{poly}(L,\log m)},$$

and for every $\overrightarrow{\mathbf{W}}'\in(\mathbb{R}^{m\times m})^L$ *with*

$$\|\overrightarrow{\mathbf{W}}'\|_2\le\frac{1}{\mathsf{poly}(L,\log m)},$$

the following inequality holds

$$\begin{aligned}F(\overrightarrow{\mathbf{W}}+\overrightarrow{\mathbf{W}}')\le{}&F(\overrightarrow{\mathbf{W}})+\langle\nabla F(\overrightarrow{\mathbf{W}}),\overrightarrow{\mathbf{W}}'\rangle\\&+O\big(\frac{nL^2m}{d}\big)\|\overrightarrow{\mathbf{W}}'\|_2^2\\&+\frac{\mathsf{poly}(L)\sqrt{nm\log m}}{\sqrt{d}}\cdot\|\overrightarrow{\mathbf{W}}'\|_2\big(F(\overrightarrow{\mathbf{W}})\big)^{1/2}\end{aligned}$$

Quite different from classical smoothness, we still have a first-order term $\|\overrightarrow{\mathbf{W}}'\|_2$ on the right hand side, but classical smoothness only has a second-order term $\|\overrightarrow{\mathbf{W}}'\|_2^2$. As one can see in our final proofs, as m goes larger (so when we over-parameterize), the effect of the first-order term becomes smaller and smaller comparing to the second-order term. This brings Theorem 4 closer and closer, but still not identical, to the classical Lipschitz smoothness.

The derivation of our main Theorem 1 and 2 from technical Theorem 3 and 4 is quite straightforward, and can be found in Section F and G.

Remark. In our proofs, we show that GD and SGD can converge fast enough and thus the weights stay close to random initialization by spectral norm bound $\frac{1}{\mathsf{poly}(n,L,\delta^{-1})}$. (This ensures Theorem 3 and 4 both apply.) This bound seems extremely small, but in fact, is large enough to to-

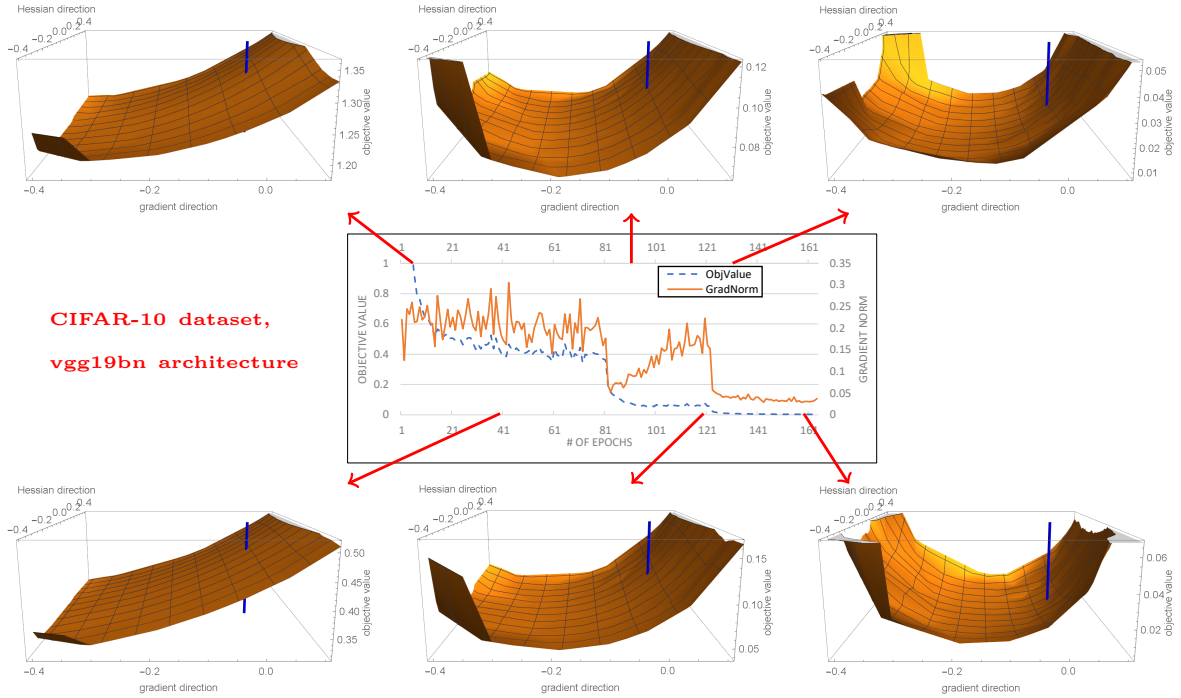

Figure 1: Landscapes of the CIFAR10 image-classification training objective $F(W)$ near points $W = W_t$ on the SGD training trajectory. The x and y axes represent the gradient direction $\nabla F(W_t)$ and the most negatively curved direction of the Hessian after smoothing (approximately found by Oja's method (Allen-Zhu & Li, 2017; 2018)). The z axis represents the objective value.

Observation. As far as minimizing objective is concerned, the (negative) gradient direction sufficiently decreases the training objective. This is consistent with our main findings Theorem 3 and 4. Using second-order information gives little help.

Remark 1. Gradient norm does not tend to zero because cross-entropy loss is not strongly convex (see Section 5).

Remark 2. The task is CIFAR10 (for CIFAR100 or CIFAR10 with noisy label, see Figure 2 through 7 in appendix).

Remark 3. Architecture is ResNet with 32 layers (for VGG19 or ResNet-110, see Figure 2 through 7 in appendix).

Remark 4. The six plots correspond to epoch 5, 40, 90, 120, 130 and 160. We start with learning rate 0.1, and decrease it to 0.01 at epoch 81, and to 0.001 at epoch 122. SGD with momentum 0.9 is used. The training code is unchanged from (Yang, 2018) and we only write new code for plotting such landscapes.

tally change the outputs and fit the training data, because weights are randomly initialized (per entry) at around $\frac{1}{\sqrt{m}}$ for m being large.

In practice, we acknowledge that one often goes beyond this theory-predicted spectral-norm boundary. However, quite interestingly, we still observe Theorem 3 and 4 happen in practice at least for image classification tasks. In Figure 1, we show the typical landscape near a point $\overrightarrow{\mathbf{W}}$ on the SGD training trajectory. The gradient is sufficiently large and going in its direction can indeed decrease the objective; in contrast, though the objective is non-convex, the negative curvature of its "Hessian" is not significant comparing to gradient. From Figure 1 we also see that the objective function is sufficiently smooth (at least in the two interested dimensions that we plot).

4 Main Techniques

Proof to Theorem 3 and 4 consist of the following steps.

Step 1: properties at random initialization. Let $\overrightarrow{\mathbf{W}} = \overrightarrow{\mathbf{W}}^{(0)}$ be at random initialization and $h_{i,\ell}$ and $\mathbf{D}_{i,\ell}$ be defined with respect to $\overrightarrow{\mathbf{W}}$. We first show that forward propagation neither explode or vanish. That is,

$$\|h_{i,\ell}\| \approx 1 \text{ for all } i \in [n] \text{ and } \ell \in [L].$$

This is basically because for a fixed y, we have $\|\mathbf{W}y\|^2$ is around 2, and if its signs are sufficiently random, then ReLU activation kills half of the norm, that is $\|\phi(\mathbf{W}y)\| \approx 1$. Then applying induction finishes the proof.

Analyzing forward propagation is not enough. We also need spectral norm bounds on the backward matrix

$$\|\mathbf{B}\mathbf{D}_{i,L}\mathbf{W}_L \cdots \mathbf{D}_{i,a}\mathbf{W}_a\|_2 \leq O(\sqrt{m/d}) \ ,$$

and on the intermediate matrix

$$\|\mathbf{D}_{i,a}\mathbf{W}_a \cdots \mathbf{D}_{i,b}\mathbf{W}_b\|_2 \leq O(\sqrt{L})$$

for every $a, b \in [L]$. Note that if one naively bounds the spectral norm by induction, then $\|\mathbf{D}_{i,a}\mathbf{W}_a\|_2 \approx 2$ and it will *exponentially blow up!* Our careful analysis ensures that even when L layers are stacked together, there is no exponential blow up in L.

The final lemma in this step proves that, as long as $\|x_i -$

$x_j\| \geq \delta$, then

$$\|h_{i,\ell} - h_{j,\ell}\| \geq \Omega(\delta) \text{ for each layer } \ell \in [L].$$

This can be proved by a careful induction. Details are in Section A.

Step 2: stability after adversarial perturbation. We show that for every $\overrightarrow{\mathbf{W}}$ that is "close" to initialization, meaning $\|\mathbf{W}_\ell - \mathbf{W}_\ell^{(0)}\|_2 \leq \omega$ for every ℓ and for some $\omega \leq \frac{1}{\mathsf{poly}(L)}$, then

(a) the number of sign changes $\|\mathbf{D}_{i,\ell} - \mathbf{D}_{i,\ell}^{(0)}\|_0$ is at most $O(m\omega^{2/3}L)$, and

(b) the perturbation amount $\|h_{i,\ell} - h_{i,\ell}^{(0)}\| \leq O(\omega L^{5/2})$.

We call this "forward stability", and it is the most technical proof of this paper.

Remark. Intuitively, both "(a) implies (b)" and "(b) implies (a)" are not hard to prove. If the number of sign changes is bounded in all layers, then $h_{i,\ell}$ and $h_{i,\ell}^{(0)}$ cannot be too far away by applying matrix concentration; and reversely, if $h_{i,\ell}$ is not far from $h_{i,\ell}^{(0)}$ in all layers, then the number of sign changes per layer must be small. Unfortunately, one cannot apply such derivation with induction, because constants will blow up exponentially in the number of layers.

Remark. In the final proof, $\overrightarrow{\mathbf{W}}$ is a point obtained by GD/SGD starting from $\overrightarrow{\mathbf{W}}^{(0)}$, and thus $\overrightarrow{\mathbf{W}}$ may depend on the randomness of $\overrightarrow{\mathbf{W}}^{(0)}$. Since we cannot control how such randomness correlates, we argue for the above two properties against *all possible* $\overrightarrow{\mathbf{W}}$.

Another main result in this step is to show that the backward matrix $\mathbf{B}\mathbf{D}_{i,L}\mathbf{W}_L \cdots \mathbf{D}_{i,a}\mathbf{W}_a$ does not change by more than $O(\omega^{1/3}L^2\sqrt{m/d})$ in spectral norm. Recall that in the Step 1 we shown that this matrix is of spectral norm $O(\sqrt{m/d})$; thus as long as $\omega^{1/3}L^2 \ll 1$, this change is somewhat negligible. Details are in Section B.

Step 3: gradient bound. The hard part of Theorem 3 is to show gradient lower bound. For this purpose, recall from Fact 2.6 that each sample $i \in [n]$ contributes to the full gradient matrix by $\mathbf{D}_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)h_{i,\ell-1}^\top$, where the backward matrix is applied to a loss vector loss_i. To show this is large, intuitively, one wishes to show $(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)$ and $h_{i,\ell-1}$ are both vectors with large Euclidean norm.

Thanks to Step 1 and 2, this is not hard for a *single* sample $i \in [n]$. For instance, $\|h_{i,\ell-1}^{(0)}\| \approx 1$ by Step 1 and we know $\|h_{i,\ell-1} - h_{i,\ell-1}^{(0)}\| \leq o(1)$ from Step 2. One can also argue for $\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i$ but this is a bit harder. Indeed, when moving from random initialization $\overrightarrow{\mathbf{W}}^{(0)}$ to $\overrightarrow{\mathbf{W}}$, the loss vector loss_i can change completely. Fortunately, $\mathsf{loss}_i \in \mathbb{R}^d$ is a low-dimensional vector, so one can calculate $\|\mathsf{Back}_{i,\ell+1}^\top u\|$ for every fixed u and then apply ε-net.

Finally, how to combine the above argument with multiple samples $i \in [n]$? These matrices are clearly not independent and may (in principle) sum up to zero. To deal with this, we use $\|h_{i,\ell} - h_{j,\ell}\| \geq \Omega(\delta)$ from Step 1. In other words, even if the contribution matrix $\mathbf{D}_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)h_{i,\ell-1}^\top$ with respect to one sample i is fixed, the contribution matrix with respect to other samples $j \in [n] \setminus \{i\}$ are still sufficiently random. Thus, the final gradient matrix will still be large. This idea comes from the prior work (Li & Liang, 2018), and helps us prove Theorem 3. Details in Appendix C and D.

Step 4: smoothness. In order to prove Theorem 4, one needs to argue, if we are currently at $\breve{\overrightarrow{\mathbf{W}}}$ and perturb it by $\overrightarrow{\mathbf{W}}'$, then how much does the objective change in second and higher order terms. This is different from our stability theory in Step 2, because Step 2 is regarding having a perturbation on $\overrightarrow{\mathbf{W}}^{(0)}$; in contrast, in Theorem 4 we need a (small) perturbation $\overrightarrow{\mathbf{W}}'$ on top of $\breve{\overrightarrow{\mathbf{W}}}$, which may already be a point perturbed from $\overrightarrow{\mathbf{W}}^{(0)}$. Nevertheless, we still manage to show that, if $\breve{h}_{i,\ell}$ is calculated on $\breve{\overrightarrow{\mathbf{W}}}$ and $h_{i,\ell}$ is calculated on $\breve{\overrightarrow{\mathbf{W}}} + \overrightarrow{\mathbf{W}}'$, then $\|h_{i,\ell} - \breve{h}_{i,\ell}\| \leq O(L^{1.5})\|\mathbf{W}'\|_2$. This, along with other properties to prove, ensures semi-smoothness. This explains Theorem 4 and details are in Section E.

Remark. In other words, the amount of changes to each hidden layer (i.e., $h_{i,\ell} - \breve{h}_{i,\ell}$) is proportional to the amount of perturbation $\|\mathbf{W}'\|_2$. This may sound familiar to some readers: a ReLU function is Lipschitz continuous $|\phi(a) - \phi(b)| \leq |a - b|$, and composing Lipschitz functions still yield Lipschitz functions. What is perhaps surprising here is that this "composition" does not create exponential blow-up in the Lipschitz continuity parameter, as long as the amount of over-parameterization is sufficient and $\breve{\overrightarrow{\mathbf{W}}}$ is close to initialization.

5 Notable Extensions

Our Step 1 through Step 4 in Section 4 in fact give rise to a general plan for proving the training convergence of any neural network (at least with respect to the ReLU activation). Thus, it is expected that it can be generalized to many other settings. Not only we can have different number of neurons each layer, our theorems can be extended at least in the following three major directions.[4]

Different loss functions. There is absolutely no need to restrict only to ℓ_2 regression loss. We prove in Appendix H

[4] In principle, each such proof may require a careful rewriting of the main body of this paper. We choose to sketch only the proof difference (in the appendix) in order to keep this paper short. If there is sufficient interest from the readers, we can consider adding the full proofs in the future revision of this paper.

that, for any Lipschitz-smooth loss function f:

Theorem 5 (arbitrary loss). *From random initialization, with probability at least $1-e^{-\Omega(\log^2 m)}$, gradient descent with appropriate learning rate satisfy the following.*

- *If f is nonconvex but σ-gradient dominant (a.k.a. Polyak-Łojasiewicz), GD finds ε-error minimizer in*[5]
$$T=\widetilde{O}\big(\tfrac{\mathsf{poly}(n,L)}{\sigma\delta^2}\cdot\log\tfrac{1}{\varepsilon}\big)\ \textit{iterations}$$
as long as $m\geq\widetilde{\Omega}\big(\mathsf{poly}(n,L,\delta^{-1})\cdot d\sigma^{-2}\big)$.
- *If f is convex, then GD finds ε-error minimizer in*
$$T=\widetilde{O}\big(\tfrac{\mathsf{poly}(n,L)}{\delta^2}\cdot\tfrac{1}{\varepsilon}\big)\ \textit{iterations}$$
as long as $m\geq\widetilde{\Omega}\big(\mathsf{poly}(n,L,\delta^{-1})\cdot d\log\varepsilon^{-1}\big)$.
- *If f is non-convex, then SGD finds a point with $\|\nabla f\|\leq\varepsilon$ in at most*[6]
$$T=\widetilde{O}\big(\tfrac{\mathsf{poly}(n,L)}{\delta^2}\cdot\tfrac{1}{\varepsilon^2}\big)\ \textit{iterations}$$
as long as $m\geq\widetilde{\Omega}\big(\mathsf{poly}(n,L,\delta^{-1})\cdot d\varepsilon^{-1}\big)$.
- *If f is cross-entropy for multi-label classification, then GD attains 100% training accuracy in at most*[7].
$$T=\widetilde{O}\big(\tfrac{\mathsf{poly}(n,L)}{\delta^2}\big)\ \textit{iterations}$$
as long as $m\geq\widetilde{\Omega}\big(\mathsf{poly}(n,L,\delta^{-1})\cdot d\big)$.

We remark here that the ℓ_2 loss is 1-gradient dominant so it falls into the above general Theorem 5. One can also derive similar bounds for (mini-batch) SGD so we do not repeat the statements here.

Convolutional neural networks (CNN). There are lots of different ways to design CNN and each of them may require somewhat different proofs. In Appendix I, we study the case when $\mathbf{A},\mathbf{W}_1,\dots,\mathbf{W}_{L-1}$ are convolutional while $\mathbf{W}_L$ and $\mathbf{B}$ are fully connected. We assume for notational simplicity that each hidden layer has $\mathfrak{d}$ points each with m channels. (In vision tasks, a point is a pixel). In the most general setting, these values $\mathfrak{d}$ and m can vary across layers. We prove the following theorem:

Theorem 6 (CNN). *As long as $m\geq\widetilde{\Omega}\big(\mathsf{poly}(n,L,\mathfrak{d},\delta^{-1})\cdot d\big)$, with high probability, GD and SGD find an ε-error solution for ℓ_2 regression in*
$$T=\widetilde{O}\big(\frac{\mathsf{poly}(n,L,\mathfrak{d})}{\delta^2}\cdot\log\varepsilon^{-1}\big)$$
iterations for CNN.

Of course, one can replace ℓ_2 loss with other loss functions in Theorem 5 to get different types of convergence rates. We do not repeat them here.

Residual neural networks (ResNet). There are lots of different ways to design ResNet and each of them may require somewhat different proofs. In symbols, between two layers, one may study $h_\ell=\phi(h_{\ell-1}+\mathbf{W}h_{\ell-1})$, $h_\ell=\phi(h_{\ell-1}+\mathbf{W}_2\phi(\mathbf{W}_1h_{\ell-1}))$, or even $h_\ell=\phi(h_{\ell-1}+\mathbf{W}_3\phi(\mathbf{W}_2\phi(\mathbf{W}_1h_{\ell-1})))$. Since the main purpose here is to illustrate the generality of our techniques but not to attack each specific setting, in Appendix J, we choose to consider the simplest residual setting $h_\ell=\phi(h_{\ell-1}+\mathbf{W}h_{\ell-1})$ (that was also studied for instance by theoretical work (Hardt & Ma, 2017)). With appropriately chosen random initialization, we prove the following theorem:

Theorem 7 (ResNet). *As long as $m\geq\widetilde{\Omega}\big(\mathsf{poly}(n,L,\delta^{-1})\cdot d\big)$, with high probability, GD and SGD find an ε-error solution for ℓ_2 regression in*
$$T=\widetilde{O}\big(\frac{\mathsf{poly}(n,L)}{\delta^2}\cdot\log\varepsilon^{-1}\big)$$
iterations for ResNet.

Of course, one can replace ℓ_2 loss with other loss functions in Theorem 5 to get different types of convergence rates. We do not repeat them here.

6 Conclusion

In this paper we demonstrate for state-of-the-art network architectures such as fully-connected neural networks, convolutional networks (CNN), or residual networks (ResNet), assuming there are n training samples without duplication, as long as the number of parameters is polynomial in n and L, first-order methods such as GD/SGD can find global optima of the training objective efficiently, that is, with running time only polynomially dependent on the total number of parameters of the network.

Figure 1 illustrates our main technical contribution. With the help of over-parameterization, near the GD/SGD training trajectory, there is no local minima and the objective is semi-smooth. The former means as long as the training objective is large, the objective gradient is also large. The latter means simply following the (opposite) gradient direction can sufficiently decrease the objective. They two together means GD/SGD finds global minima on over-parameterized feedforward neural networks.

There are plenty of open directions following our work, especially how to extend our result to other types of deep learning tasks and/or proving generalization. There is already generalization theory (Allen-Zhu et al., 2018a) for over-parameterized *three*-layer neural networks, so can we go any deeper?

[5] Note that the loss function when combined with the neural network together $f(\mathbf{B}h_{i,L})$ is *not* gradient dominant. Therefore, one cannot apply classical theory on gradient dominant functions to derive our same result.

[6] Again, this cannot be derived from classical theory of finding approximate saddle points for non-convex functions, because weights $\overrightarrow{\mathbf{W}}$ with small $\|\nabla f(\mathbf{B}h_{i,L})\|$ is a very different (usually much harder) task comparing to having small gradient with respect to $\overrightarrow{\mathbf{W}}$ for the entire composite function $f(\mathbf{B}h_{i,L})$.

[7] This is because attaining constant objective error $\varepsilon=1/4$ for the cross-entropy loss suffices to imply perfect training accuracy.

References

Alaeddini, A., Alemzadeh, S., Mesbahi, A., and Mesbahi, M. Linear model regression on time-series data: Non-asymptotic error bounds and applications. *arXiv preprint arXiv:1807.06611*, 2018.

Allen-Zhu, Z. and Li, Y. Follow the Compressed Leader: Faster Online Learning of Eigenvectors and Faster MMWU. In *ICML*, 2017. Full version available at `http://arxiv.org/abs/1701.01722`.

Allen-Zhu, Z. and Li, Y. Neon2: Finding Local Minima via First-Order Oracles. In *NeurIPS*, 2018. Full version available at `http://arxiv.org/abs/1711.06673`.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers. *arXiv preprint arXiv:1811.04918*, November 2018a.

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

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning (ICML)*, pp. 173–182, 2016.

Arora, S., Cohen, N., Golowich, N., and Hu, W. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018a.

Arora, S., Hazan, E., Lee, H., Singh, K., Zhang, C., and Zhang, Y. Towards provable control for unknown linear dynamical systems. 2018b.

Bartlett, P., Helmbold, D., and Long, P. Gradient descent with identity initialization efficiently learns positive definite linear transformations. In *International Conference on Machine Learning (ICML)*, pp. 520–529, 2018.

Blum, A. L. and Rivest, R. L. Training a 3-node neural network is np-complete. In *Machine learning: From theory to applications (A preliminary version of this paper was appeared in NIPS 1989)*, pp. 9–28. Springer, 1993.

Brutzkus, A. and Globerson, A. Globally optimal gradient descent for a convnet with gaussian inputs. In *International Conference on Machine Learning (ICML)*. http://arxiv.org/abs/1702.07966, 2017.

Burke, J. V., Lewis, A. S., and Overton, M. L. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

Daniely, A. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pp. 105–117. ACM, 2016.

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

Daniely, A. and Shalev-Shwartz, S. Complexity theoretic limitations on learning dnfs. In *Conference on Learning Theory (COLT)*, pp. 815–830, 2016.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. On the sample complexity of the linear quadratic regulator. *arXiv preprint arXiv:1710.01688*, 2017.

Dean, S., Tu, S., Matni, N., and Recht, B. Safely learning to control the constrained linear quadratic regulator. *arXiv preprint arXiv:1809.10121*, 2018.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, November 2018a.

Du, S. S., Lee, J. D., Tian, Y., Póczos, B., and Singh, A. Gradient descent learns one-hidden-layer CNN: don't be afraid of spurious local minima. In *International Conference on Machine Learning (ICML)*. http://arxiv.org/abs/1712.00779, 2018b.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. *ArXiv e-prints*, 2018c.

Fung, V. An overview of resnet and its variants. *https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035*, 2017.

Ge, R., Lee, J. D., and Ma, T. Learning one-hidden-layer neural networks with landscape design. In *ICLR*, 2017. URL `http://arxiv.org/abs/1711.00501`.

Goel, S., Kanade, V., Klivans, A., and Thaler, J. Reliably learning the ReLU in polynomial time. In *Conference on Learning Theory (COLT)*, 2017.

Goel, S., Klivans, A., and Meka, R. Learning one convolutional layer with overlapping patches. In *International Conference on Machine Learning (ICML)*. arXiv preprint arXiv:1802.02547, 2018.

Goodfellow, I. J., Vinyals, O., and Saxe, A. M. Qualitatively characterizing neural network optimization problems. In *ICLR*, 2015.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6645–6649. IEEE, 2013.

Hardt, M. and Ma, T. Identity matters in deep learning. In *ICLR*, 2017. URL http://arxiv.org/abs/1611.04231.

Hardt, M., Ma, T., and Recht, B. Gradient descent learns linear dynamical systems. *Journal of Machine Learning Research (JMLR)*, 19(29):1–44, 2018.

Hazan, E., Singh, K., and Zhang, C. Learning linear dynamical systems via spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6702–6712, 2017.

Hazan, E., Lee, H., Singh, K., Zhang, C., and Zhang, Y. Spectral filtering for general linear dynamical systems. In *Advances in Neural Information Processing Systems (NINPS)*, 2018.

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.

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

Klivans, A. R. and Sherstov, A. A. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Li, Y. and Yuan, Y. Convergence analysis of two-layer neural networks with ReLU activation. In *Advances in Neural Information Processing Systems (NeurIPS)*. http://arxiv.org/abs/1705.09886, 2017.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Livni, R., Shalev-Shwartz, S., and Shamir, O. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 855–863, 2014.

Manurangsi, P. and Reichman, D. The computational complexity of training ReLU(s). *arXiv preprint arXiv:1810.04207*, 2018.

Marecek, J. and Tchrakian, T. Robust spectral filtering and anomaly detection. *arXiv preprint arXiv:1808.01181*, 2018.

Nesterov, Y. *Introductory Lectures on Convex Programming Volume: A Basic course*, volume I. Kluwer Academic Publishers, 2004. ISBN 1402075537.

Oymak, S. Learning compact neural networks with regularization. *arXiv preprint arXiv:1802.01223*, 2018.

Oymak, S. and Ozay, N. Non-asymptotic identification of LTI systems from a single trajectory. *arXiv preprint arXiv:1806.05722*, 2018.

Panigrahy, R., Rahimi, A., Sachdeva, S., and Zhang, Q. Convergence results for neural networks via electrodynamics. In *ITCS*, 2018.

Safran, I. and Shamir, O. Spurious local minima are common in two-layer ReLU neural networks. In *International Conference on Machine Learning (ICML)*. http://arxiv.org/abs/1712.08968, 2018.

Shamir, O. A variant of azuma's inequality for martingales with subgaussian tails. *ArXiv e-prints*, abs/1110.2392, 10 2011.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

Simchowitz, M., Mania, H., Tu, S., Jordan, M. I., and Recht, B. Learning without mixing: Towards a sharp analysis of linear system identification. In *Conference on Learning Theory (COLT)*. arXiv preprint arXiv:1802.08334, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Soltanolkotabi, M. Learning ReLUs via gradient descent. *CoRR*, abs/1705.04591, 2017. URL http://arxiv.org/abs/1705.04591.

Song, L., Vempala, S., Wilmes, J., and Xie, B. On the complexity of learning neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5514–5522, 2017.

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

Srivastava, R. K., Greff, K., and Schmidhuber, J. Training very deep networks. In *Advances in neural information processing systems (NeurIPS)*, pp. 2377–2385, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

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 (ICML)*. http://arxiv.org/abs/1703.00560, 2017.

Yang, W. Classification on CIFAR-10/100 and ImageNet with PyTorch, 2018. URL `https://github.com/bearpaw/pytorch-classification`. Accessed: 2018-04.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Zhang, J., Lin, Y., Song, Z., and Dhillon, I. S. Learning long term dependencies via Fourier recurrent units. In *International Conference on Machine Learning (ICML)*. arXiv preprint arXiv:1803.06585, 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. In *International Conference on Machine Learning (ICML)*. arXiv preprint arXiv:1706.03175, 2017b.