Early Stage Convergence and Global Convergence of Training Mildly Parameterized Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

The convergence of GD and SGD when training mildly parameterized neural networks starting from random initialization is studied. For a broad range of models and loss functions, including the most commonly used square loss and cross entropy loss, we prove an "early stage convergence" result. We show that the loss is decreased by a significant amount in the early stage of the training, and this decrease is fast. Furthurmore, for exponential type loss functions, and under some assumptions on the training data, we show global convergence of GD. Instead of relying on extreme over-parameterization, our study is based on a microscopic analysis of the activation patterns for the neurons, which helps us derive more powerful lower bounds for the gradient. The results on activation patterns, which we call "neuron partition", help build intuitions for understanding the behavior of neural networks' training dynamics, and may be of independent interest.

1 Introduction

As deep learning shows its capability in various fields of applications, extensive researches are done to theoretically understand and explain its great success. Among the topics covered by these theoretical studies, the optimization of deep neural networks is one of the most crucial one, especially given the fact that simple optimization algorithms such as Gradient Descent (GD) and Stochastic Gradient Descent (SGD) can easily achieve zero training loss (Zhang et al., 2017), although the loss landscape of training neural networks is highly non-convex (Li et al., 2018; Sun et al., 2020). Existing works attempting to answer the surprising convergence ability usually work in settings that do not align well with realistic practices. For example, the notable Neural Tangent Kernel (NTK) theory (Jacot et al., 2018; Du et al., 2018; 2019; Allen-Zhu et al., 2019b; Zou et al., 2018) shows the global convergence of SGD in highly over-parameterized regime, in which the loss landscape is approximated by a quadratic function. In practice, though, the loss landscape is indeed highly non-convex with spurious local minima and saddle points (Safran and Shamir, 2018; Safran et al., 2021), and the dynamics of neurons show nonlinear behaviors (Ma et al., 2020).

Nevertheless, fast decreasing of the loss value always happens when the network trained is not highly over-parameterized, at least in the early stage of training. It is common that the loss experiences a drastic decreasing at the beginning of the training. In many cases, this decrease of loss even continues until the loss achieves zero, i.e. the optimization algorithm fully converges. In this paper, we study the early stage or full convergence of GD and SGD when under-parameterized or mildly over-parameterized models are trained. Specifically, we answer the following two theoretical questions:

When we train practical-size neural networks by GD or SGD,

1. Does the fast convergence in the early stage of the training provably exist? If so, how

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

long will the phenomenon last and how much will loss descend in the early stage?

2. Can the global convergence be proved under some special conditions on loss function and training data?

Our answers to the questions above are roughly summarized in the following main theorems:

Main Theorem 1 (Informal statement of Theorem 4.2 and 4.6)**.** *For mildly over-parameterized or under-parameterized two-layer neural networks with quadratic loss or general classification loss, let parameters be trained by GD or SGD started with random initialization. If the learning rate $\eta \lesssim 1$, then in the first $\Theta(1/\eta)$ iterations, the loss will descend $\Omega(1)$.*

Main Theorem 2 (Informal statement of Theorem 5.2 and 5.3)**.** *For mildly over-parameterized or under-parameterized two-layer neural networks with exponential-type loss, let parameters be trained by GD with random initialization and proper learning rate. If data is well-separated, the loss converges to 0 at exponential rate or arbitrary polynomial rate, depending on the conditions.*

In the first main theorem, we demonstrate that the fast convergence in the early stage of training happens under weak conditions. The model does not need to be highly over-parameterized, and the loss function can take quite general forms. In the second main theorem, we further prove the full convergence of GD for exponential-type loss and well-separated dataset. These assumptions on the loss function and training data are close to practice and widely picked by previous theoretical studies (Ji and Telgarsky, 2018; 2020; Phuong and Lampert, 2020; Chatterji et al., 2021b;a; Lyu and Li, 2019; Lyu et al., 2021). "Full convergence" means that our analysis covers all the stages of the training process, starting from initialization to the convergence to zero loss. This is different from existing works which focus on the convergence process after the training accuracy hits 100% (Lyu and Li, 2019; Ji and Telgarsky, 2020; Chatterji et al., 2021b). Moreover, we provide the convergence rate of GD for two classes of neural networks.

During the theoretical analysis, we study the dynamics of neurons in detail and capture the effect of each sample on each neuron.. We call our results "neuron partition", which provides a accurate description to the behavior of neurons. With the neuron partition results, we derive a novel gradient lower bound for mildly parameterized neural networks, which is vital for the convergence results. The neuron partition also provides rich intuition for how the convergence happens. When GD or SGD is used, at the initial stage of training, the loss landscape may be complicated. However, neurons will adjust their directions rapidly and enter a good region which contains neither spurious local minima nor saddle points. Then, neurons will keep moving through the right directions for $\Theta(1/\eta)$ iterations, during which the loss descends quickly and significantly. For exponential-type loss and well-separated training data, the second stage will proceed uninterruptedly until training is stopped.

2 Related Work

Due to the highly non-convex loss landscape (Li et al., 2018; Sun et al., 2020), classical optimization theories such as convex optimization fail to characterize the convergence of training neural networks. Researchers have proposed theories specifically for neural networks. We list some below.

A popular line of works focus on the highly over-parameterized neural networks and derive the "Neural tanget kernel (NTK)" theory. In the NTK regime (Jacot et al., 2018; Du et al., 2018; 2019; Allen-Zhu et al., 2019a;b; Arora et al., 2019a;b; Daniely, 2017; Zou et al., 2018; Li and Liang, 2018; Chizat et al., 2019; E et al., 2019; 2020), the neural network model is close to a kernel method, leading to the nearly convex optimization landscape. Global convergence is proven in this regime. Besides, for highly over-parameterized neural networks, the mean-field approach (Chizat and Bach, 2018; Mei et al., 2018; 2019) is another line, which analyze the training dynamics by Wasserstein gradient flow.

However, the NTK regime is different from real practical neural networks in several aspects. First, practical neural networks are often mildly over-parameterized rather than highly over-parameterized (Livni et al., 2014). Second, the landscape for practical neural networks is usually more complicated, containing local minima and saddle points (Safran and Shamir, 2018; Safran et al., 2021; Ding et al., 2019). Finally, the empirical superiority of neural networks over kernel methods is obvious. For example, neural networks can learn single neuron efficiently (Yehudai and Ohad, 2020; Wu, 2022), while kernel methods fail unless the network size is exponentially large with respect to the input dimension (Yehudai and Shamir, 2019).

Furthermore, optimization theories for neural networks beyond the NTK regime has also been studied. Safran and Shamir (2018); Ding et al. (2019) proved that spurious local minima are common in the loss landscape. Safran et al. (2021) pointed out that neither one-point convexity nor Polyak-Łojasiewicz condition holds near the global minimum. In (Lyu and Li, 2019; Ji and Telgarsky, 2020; Chatterji et al., 2021b;a; Zhou et al., 2021), local convergence results are given with cross-entropy loss or some special networks at the late stage of training, i.e. when training loss is small enough. In (Li et al., 2020; Phuong and Lampert, 2020; Lyu et al., 2021; Jentzen and Riekert, 2021; Cheridito et al., 2022), global convergence results are given for some special networks, special data distribution or special target functions.

3 Preliminaries

3.1 Notations

We use capital letters for vectors or matrices and lowercase letters for scalars, e.g. $\boldsymbol{x} = (x_1, \cdots, x_d)^\top \in \mathbb{R}^d$ and $\mathbf{P} = (P_{ij})_{m_1 \times m_2} \in \mathbb{R}^{m_1 \times m_2}$. We use $\langle \cdot, \cdot \rangle$ to denote the standard Euclidean inner product between two vectors, and $\|\cdot\|$ to denote the l_2 norm of a vector or the spectral norm of a matrix. We use $a \lesssim b$ to indicate that there exists an absolute constant $c > 0$ such that $a \leq cb$, and $a \gtrsim b$ is similarly defined. We use standard progressive representation $\mathcal{O}, \Omega, \Theta$ to hide absolute constants. Let $[n] = \{1, \cdots, n\}$. Denote by $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Denote by $\mathbb{U}(S)$ the uniform distribution on a set S. Denote by $\mathbb{I}\{E\}$ the indicator function for an event E.

3.2 Problem settings

In this paper, we consider supervised learning problems. Let $\mathcal{P}$ be a data distribution on $\mathbb{R}^d \times \mathbb{R}^C$. We are given n training data $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^C$ drawn i.i.d. from $\mathcal{P}$. Without loss of generality, we assume $\|\boldsymbol{x}\| \leq 1$ for any $(\boldsymbol{x}, \boldsymbol{y})$ sampled from $\mathcal{P}$.

We consider the empirical risk minimization (ERM) problem, which tries to minimize the empirical risk $\mathcal{L}(\cdot)$ with loss function $\ell(\cdot, \cdot)$:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i, \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\theta})), \tag{1}$$

where $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^C$ is the model and $\boldsymbol{\theta}$ represents all parameters of the model. We will give specific forms to the loss function and the model in the analysis in later sections.

We use Gradient Descent (GD) or Stochastic Gradient Descent (SGD) starting from random initialization to solve the ERM problem. The update rules of GD or batch SGD can be written as:

$$\textbf{GD}: \quad \boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta_t \nabla \mathcal{L}(\boldsymbol{\theta}(t)), \tag{2}$$

$$\textbf{SGD}: \quad \boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \frac{\eta_t}{B} \sum_{i \in \mathcal{B}_t} \nabla \ell(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\theta}(t))), \tag{3}$$

where $\mathcal{B}_t = \{\gamma_{t,1}, \cdots, \gamma_{t,B}\}$ is a batch, and $\gamma_{t,1}, \cdots, \gamma_{t,B} \overset{\text{i.i.d.}}{\sim} \mathbb{U}([n])$ and are independent with $\boldsymbol{\theta}(t)$. The random initialization will be specified in each theorem.

4 Early Stage Convergence

In this section, we state and discuss our main results on the fast convergence in the early stage of training. The models we focus on are mildly over-parameterized or under-parameterized neural networks. As a warm up, in Section 4.1, we study a simple case with binary classification problems and quadratic loss. Then, in Section 4.2, we extend our results to multi-class classification problems with general loss and one-hot labels. We provide discussions to our results in Section 4.3.

4.1 Binary classification with quadratic loss

In this subsection, we study the ERM problem (1) with quadratic loss $\ell(y_1, y_2) = \frac{1}{2}(y_1 - y_2)^2$ and the two-layer ReLU neural network model without bias:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{k=1}^{m} a_k \sigma(\boldsymbol{b}_k^\top \boldsymbol{x}), \tag{4}$$

where $\sigma(z) = \mathrm{ReLU}(z)$, $\boldsymbol{\theta} = (a_1, \cdots, a_m, \boldsymbol{b}_1^\top, \cdots, \boldsymbol{b}_m^\top)^\top \in \mathbb{R}^{dm+m}$. We consider the following random initialization

$$a_k(0) \overset{\text{i.i.d.}}{\sim} \mathbb{U}(\pm 1/\sqrt{m}), \quad \boldsymbol{b}_k(0) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\boldsymbol{0}, \frac{\kappa^2}{md}\mathbf{I}_d), \quad \text{for } k \in [n], \tag{5}$$

where κ is a constant that controls the initial scale of the first layer.

We focus on the training data with two classes with good separation given by the following assumption.

Assumption 4.1. n is even. $y_i = 1$ for $i \in [n/2]$; $y_i = -1$ for $i \in [n] - [n/2]$. $\boldsymbol{x}_i^\top \boldsymbol{x}_j \geq 0$ for i, j in the same class; $\boldsymbol{x}_i^\top \boldsymbol{x}_j \leq 0$ for i, j in different classes.

Similar assumption has been used in prior works (Phuong and Lampert, 2020). For image classification problems, this assumption can be ensured by a simple transformation on the data due to the non-negative pixels of images. Specifically, given an image dataset consisted of two classes, we just need to replace $\boldsymbol{x}$ with $-\boldsymbol{x}$ for any $(\boldsymbol{x}, y)$ in the second class.

Now we state the following result.

Theorem 4.2. *Suppose Assumption 4.1 holds. Let $\boldsymbol{\theta}(t)$ be the parameters of model (4) trained by Gradient Descent (2) with quadratic loss. If the width $m = \Omega(\log(n/\delta))$, the input dimension $d = \Omega(\log m)$, the learning rate $\eta_t = \eta \lesssim 1$ and the initialization scale $\kappa = \mathcal{O}(\eta/n)$ in (5), then, with probability at least $1 - \delta - 2me^{-2d}$, the loss will descend $\Omega(1)$ in $T = \Theta(\frac{1}{\eta})$ iterations.*

From our proof in Appendix A, we have the following corollary with fixed numbers about η, κ, m instead of progressive expressions.

Corollary 4.3. *Suppose Assumption 4.1 holds. Let $\boldsymbol{\theta}(t)$ be the parameters of model (4) trained by Gradient Descent (2) with quadratic loss. Let $\eta = \mathbf{0.01}$, $\kappa \leq \min\{\eta/2000, \eta/3n\}$, $m \geq \max\{144\log(2n^2/\delta), 4\}$, and $\gamma_1, \gamma_2 = \Theta(1)$ be constants related with the data distribution defined in (10) in Appendix A. Then, with probability at least $1 - \delta - 2me^{-2d}$, in $T = \mathbf{44}$ iterations, loss will descend at least $0.193(\gamma_1 - \gamma_2\sqrt{\log(2n^2/\delta)/m}) = \Omega(1 - \sqrt{\log(n/\delta)/m})$.*

4.2 Multi-class classification with general loss and one-hot labels

In this subsection, we extend the early stage convergence results above to more practical settings. Specifically, we consider classification problem with one-hot labels and the general loss functions that satisfy the following conditions.

Assumption 4.4. The loss function $\ell(\cdot, \cdot)$ can be expressed as $\ell(\boldsymbol{y}_1, \boldsymbol{y}_2) = \tilde{\ell}(\boldsymbol{y}_1^\top \boldsymbol{y}_2)$ such that: (i) $\tilde{\ell}(\cdot)$ is twice differentiable in $\mathbb{R}$; (ii) $\tilde{\ell}(\cdot)$ is non-negative and non-increasing in $\mathbb{R}$; (iii) there exist $z_0 \in (0, 1]$, $g_{\min}, g_{\max} > 0, h_{\max} \geq 0$ such that $g_{\min} \leq -\tilde{\ell}'(z) \leq g_{\max}$ and $0 \leq \tilde{\ell}''(z) \leq h_{\max}$ hold for any $z \in [0, z_0]$.

It is easy to verify that exponential loss $\ell(\boldsymbol{y}_1, \boldsymbol{y}_2) = e^{-\boldsymbol{y}_1^\top \boldsymbol{y}_2}$ ($z_0 = 1$, $g_{\min} = \frac{1}{e}$, $g_{\max} = 1$, $h_{\max} = 1$), logistic loss $\ell(\boldsymbol{y}_1, \boldsymbol{y}_2) = \log(1 + e^{-\boldsymbol{y}_1^\top \boldsymbol{y}_2})$ ($z_0 = 1$, $g_{\min} = \frac{1}{e+1}$, $g_{\max} = \frac{1}{2}$, $h_{\max} = \frac{1}{4}$) and hinge loss $\ell(\boldsymbol{y}_1, \boldsymbol{y}_2) = \max\{0, 1 - \boldsymbol{y}_1^\top \boldsymbol{y}_2\}$ ($z_0 = 1$, $g_{\min} = 1$, $g_{\max} = 1$, $h_{\max} = 0$) all satisfy the conditions in Assumption 4.4.

On the training data, we impose the following assumption, which is milder than Assumption 4.1.

Assumption 4.5. There exists $s > -1$ s.t. $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \geq s$ for any $i, j \in [n]$.

Note that now we do not need any separability of training data, but only require there is no pair of data going into opposite directions. (We can even normalize the data and let $\|\boldsymbol{x}\| < 1 - \epsilon$ for some $\epsilon > 0$, rather than $\|\boldsymbol{x}\| \leq 1$. Then, the above assumption naturally holds.)

For the model, we use the two-layer ReLU neural network model with bias as our prediction model:

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{m} \boldsymbol{a}_k \sigma(\boldsymbol{b}_k^\top \boldsymbol{x} + c_k), \tag{6}$$

where $\sigma(z) = \mathrm{ReLU}(z)$, $\boldsymbol{a}_k \in \mathbb{R}^C$ and $\boldsymbol{\theta} = (\boldsymbol{a}_1^\top, \cdots, \boldsymbol{a}_m^\top, \boldsymbol{b}_1^\top, \cdots, \boldsymbol{b}_m^\top, c_1, \cdots, c_m)^\top \in \mathbb{R}^{(d+C+1)m}$. We consider random initialization $(\boldsymbol{b}_k(0)^\top, c_k(0))^\top \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \frac{\kappa^2}{m(d+1)}\mathbf{I}_d)$ and $\boldsymbol{a}_k(0) = (1/\sqrt{m}, \cdots, 1/\sqrt{m})^\top$ for $k \in [m]$, where κ is again a constant that controls the initial scale of the first layer.

We show the following convergence the result:

Theorem 4.6. *Under Assumption 4.4 and 4.5, let $\boldsymbol{\theta}(t)$ be the parameters of model (6) trained by GD (2) or SGD (3). If the width $m = \Omega(\log(n/\delta))$, the input dimension $d = \Omega(\log m)$, the learning rate $\eta_t = \eta \lesssim 1$ and the initialization scale $\kappa = \mathcal{O}(\eta)$, then with probability at least $1 - \delta - \mathcal{O}(me^{-d})$, the loss will descend $\Omega(1)$ in $T = \Theta(\frac{1}{\eta})$ iterations.*

By our analysis in Appendix B, we have the following corollary which gives specific numbers about η, κ, m instead of progressive expression.

Corollary 4.7. *Under the same assumptions as Theorem 4.6, let $\boldsymbol{\theta}(t)$ be the parameters of the model (6) trained by GD (2) or SGD (3). Let $\eta = \mathbf{0.01}$, $\kappa \le \eta/2000$, $m \ge \max\{13\log(2n^2/\delta), 6\}$; $z_0 = 1$, $g_{\min} = \frac{1}{2}$, $g_{\max} = 1$, $h_{\max} = 1$ in Assumption 4.4 and $s = 0$ in Assumption 4.5. Then, with probability at least $1 - \delta - 4me^{-\frac{d+1}{2}}$, in $T = \mathbf{34}$ iterations, loss will descend at least $0.34(0.2176 - \sqrt{\log(2n^2/\delta)/2m}) = \Omega(1 - \sqrt{\log(n/\delta)/m})$.*

4.3 Discussion

A large number of classical convergence analysis focus on the late stage of training. For neural networks' optimization, many works focus on the period of training after the loss is smaller enough (Lyu and Li, 2019; Ji and Telgarsky, 2020; Chatterji et al., 2021b; Zhou et al., 2021). However, the early stage is at least as important as the late stage, especially for non-convex problems.

The theoretical results we present in this section demonstrate the fast convergence during the early stage of optimization, i.e. the (short) period of time after the initialization. The two main results, Theorem 4.2 and 4.6, imply that if we train two-layer neural networks by GD or SGD with learning rate $\eta \lesssim 1$, then the loss will descend significantly ($\Omega(1)$) in the first $T = \Theta(1/\eta)$ iterations.

Our results work under realistic settings. For models, our results hold for mildly over-parameterized and under-parameterized neural networks, because we only need the width $m = \Omega(\log(n/\delta))$. For loss functions, our results apply for practical losses, such as quadratic loss and cross entropy loss. On the algorithm side, the results work for both GD (2) and SGD (3).

Though both concerning convergence of GD and SGD, our results are essentially different from the NTK theory, especially on the requirement of over-parameterization. Our results only need $m = \Omega(\log(n/\delta))$, while in the NTK analysis usually assumes that m is polynomial in $n, 1/\kappa$ or etc, excluding the practical settings. Moreover, from the analysis side, we establish fine-grained analysis of the training dynamics of each neuron, which may be of independent interested. See Section 6 for a summary of our techniques, and Appendix A and B for the detailed proof. Our analysis helps understand how the convergence in the early stage happens. When GD or SGD is used, the loss landscape may be complicated near the random initialization. However, neurons will adjust directions rapidly, and the iterator will enter a good region which contains neither spurious local minima nor saddle points. Then, neurons will keep going towards the right directions for a period of time, during which process the loss will descend fast and significantly.

5 Global Convergence

When stronger conditions are imposed on the data distribution and the loss function, we can show global convergence of GD, i.e. starting from any initialization, the GD descends the loss to 0.

For loss functions, we consider the following exponential-type classification loss:

Assumption 5.1. The loss function $\ell(\cdot,\cdot)$ can be expressed as $\ell(\boldsymbol{y}_1,\boldsymbol{y}_2)=\tilde{\ell}(\boldsymbol{y}_1^\top\boldsymbol{y}_2)$ such that: (i) $\tilde{\ell}(\cdot)$ is twice continuously differentiable in $\mathbb{R}$; (ii) $\tilde{\ell}(\cdot)$ is positive and non-increasing in $\mathbb{R}$; (iii) there exist $g_a,g_b>0,h\geq 0$ s.t. $g_a\leq -\frac{\tilde{\ell}'(z)}{\tilde{\ell}(z)}\leq g_b$ and $0\leq\frac{\tilde{\ell}''(z)}{\tilde{\ell}(z)}\leq h$ for any $z\geq 0$.

The similar assumption on exponential-type loss has been used in prior works (Lyu and Li, 2019). It is easy to verify that both exponential loss $\ell(\boldsymbol{y}_1,\boldsymbol{y}_2)=e^{-\boldsymbol{y}_1^\top\boldsymbol{y}_2}$ ($g_a=g_b=h=1$) and logistic loss $\ell(\boldsymbol{y}_1,\boldsymbol{y}_2)=\log(1+e^{-\boldsymbol{y}_1^\top\boldsymbol{y}_2})$ ($g_a=\frac{1}{2},g_b=1,h=1$) satisfy the conditions in Assumption 5.1 above.

Let V be a constant defined as

$$V=\frac{1}{32}\Big(\frac{1}{8}-\sqrt{\frac{\log(n^2/\delta)}{2m}}\Big)\max\Big\{\frac{2}{n}+\frac{n-2}{n}\gamma,\lambda_{\min}(\mathbf{X}_+^\top\mathbf{X}_+)\wedge\lambda_{\min}(\mathbf{X}_-^\top\mathbf{X}_-)\Big\},\qquad(7)$$

where $\mathbf{X}_+:=(\boldsymbol{x}_1,\cdots,\boldsymbol{x}_{\frac{n}{2}})\in\mathbb{R}^{d\times\frac{n}{2}}$, $\mathbf{X}_-:=(\boldsymbol{x}_{\frac{n}{2}+1},\cdots,\boldsymbol{x}_n)\in\mathbb{R}^{d\times\frac{n}{2}}$, and $\gamma:=\min_{i,j\text{in the same class}}\boldsymbol{x}_i^\top\boldsymbol{x}_j\geq 0$. We have the following convergence theorems in which V controls the convergence speed.

Theorem 5.2. *Under Assumption 5.1 and Assumption 4.1, let $\boldsymbol{\theta}(t)$ be the parameters of model (4) trained by Gradient Descent (2) starting from random initialization (5). Let the width $m=\Omega(\log(n/\delta))$, the initialization scale $\kappa=\mathcal{O}(\eta_0/n)$ in (5), V be defined in (7), c be a constant in $(0,\frac{1}{6(1+2\eta_0)^2+2}]$, the constant $c'>0$ be sufficiently small, the hitting time $T_0=\lceil(n\log 2)^{\frac{2}{Vc}}\rceil$, the parameter $r\in[1,+\infty)$ and the learning rate satisfy*

$$\begin{cases}\eta_0\lesssim 1, & \text{for } t=0\\ \eta_t=\frac{c}{t\mathcal{L}(\boldsymbol{\theta}(t))}, & \text{for } 1\leq t<T_0\\ \eta_t=\frac{c'}{\mathcal{L}(\boldsymbol{\theta}(t))^{1-\frac{1}{2r}}}, & \text{for } t\geq T_0\end{cases}.$$

*Then, with probability at least $1-\delta-2me^{-2d}$, GD will converge at **polynomial** rate:*

$$\begin{cases}\mathcal{L}(\boldsymbol{\theta}(t))\leq\frac{\log 2}{t^{\frac{Vc}{2}}}, & 1\leq t<T_0\\ \mathcal{L}(\boldsymbol{\theta}(t))=\mathcal{O}\big(\frac{1}{t^r}\big), & t\geq T_0\end{cases}.$$

Theorem 5.3. *Under Assumption 5.1 and Assumption 4.1, let $\{\boldsymbol{b}_k(t)\}_{k\in[m]}$ be the parameters of model (4) trained by Gradient Descent (2) starting from random initialization (5). Let the width $m=\Omega(\log(n/\delta))$, the initialization scale $\kappa=\mathcal{O}(\eta_0/n)$ in (5), the constant V be defined in (7), the constant $c\leq\frac{1}{2}$ and the learning rate satisfy*

$$\begin{cases}\eta_0\lesssim 1, & \text{for } t=0\\ \eta_t=\frac{c}{\mathcal{L}(\boldsymbol{\theta}(t))}, & \text{for } t\geq 1\end{cases}.$$

*Then with probability at least $1-\delta-2me^{-2d}$, GD will converge at **exponential** rate:*

$$\mathcal{L}(\boldsymbol{\theta}(t))\leq\Big(1-\frac{Vc}{2}\Big)^{t-1}\log 2,\quad t\geq 1.$$

Theorem 5.2 and 5.3 describe the whole training process starting from random initialization to convergence. If all parameters in the network are trained, we obtain the global convergence with *arbitrary polynomial* rate. On the other hand, if only the first layer parameters $\{\boldsymbol{b}_k\}_{k\in[m]}$ are trained, we can obtain *exponential* convergence rate. Fixing some layers of the neural network is a common practice for theoretical studies (Chatterji et al., 2021b). The results show that adaptively increasing learning rate helps GD achieve faster convergence. Similar idea has been explored in previous works such as (Lyu and Li, 2019; Chatterji et al., 2021b).

A similar convergence result was proven in Phuong and Lampert (2020). where the authors showed a global convergence result of two-layer neural networks trained by Gradient Flow (GF) with cross entropy loss and orthogonal separable data. The main difference between our results and analysis with that in (Phuong and Lampert, 2020) is that we study GD instead of GF. Due to the discretization, GD is more complicated than GF. For example, the balanced property of two layers ($\sum_{k\in[m]}|a_k(t)|^2-\sum_{k\in[m]}\|\boldsymbol{b}_k(t)\|^2\equiv 0,\ \forall t\geq 0$) holds for GF throughout the training, but it does not hold for GD. Dealing with GD requires new analysis techniques and finer analysis, such as the neuron partition results we derive.

6 Proof Sketch and Techniques

6.1 Main techniques in the proof of Theorem 4.2

Inspired by the empirical work (Ma et al., 2020), we study the nonlinear behavior of each neuron in the early stage of training. First, we define the hitting time T, and we will study the convergence in the early stage $0 \le t \le T$:

$$T := \sup\Big\{t \in \mathbb{N}: \max_{i\in[n]} |f(\boldsymbol{x}_i;\boldsymbol{\theta}(s))| \le 1,\ a_k(s)a_k(0) > 0,\ \forall k \in [m],\ \forall 0 \le s \le t+1\Big\}.$$

6.1.1 Neuron partition: fine-grained dynamical analysis of each neuron and each sample

During our proof, we conduct fine-grained analysis to the interaction of neurons and samples. Specifically, we characterize the impact of each sample on each neuron, and for each sample classify neurons into four categories according to their effect. We call this analysis "neuron partition".

For the i-th training data $(\boldsymbol{x}_i, y_i)$, we divide all neurons into four categories during training $0\le t \le T$ and study them separately. We define the true-living neurons $\mathcal{TL}_i(t)$, the true-dead neurons $\mathcal{TD}_i(t)$, the false-living neurons $\mathcal{FL}_i(t)$ and the false-dead neurons $\mathcal{FD}_i(t)$ at time t as:

$$\mathcal{TL}_i(t) := \Big\{k \in [m] : y_i a_k(t) > 0, \boldsymbol{b}_k(t)^\top \boldsymbol{x}_i \ge 0\Big\},$$
$$\mathcal{TD}_i(t) := \Big\{k \in [m] : y_i a_k(t) > 0, \boldsymbol{b}_k(t)^\top \boldsymbol{x}_i < 0\Big\},$$
$$\mathcal{FL}_i(t) := \Big\{k \in [m] : y_i a_k(t) < 0, \boldsymbol{b}_k(t)^\top \boldsymbol{x}_i \ge 0\Big\},$$
$$\mathcal{FD}_i(t) := \Big\{k \in [m] : y_i a_k(t) < 0, \boldsymbol{b}_k(t)^\top \boldsymbol{x}_i < 0\Big\}.$$

It is easy to verify $[m] = \mathcal{TL}_i(t) \bigcup \mathcal{TD}_i(t) \bigcup \mathcal{FL}_i(t) \bigcup \mathcal{FD}_i(t)$ (see Lemma A.2).

Under a weak assumption on the width of the network, we have the following results for the neuron partition at initialization:

Lemma 6.1 (Informal statement of Lemma A.3)**.** *If $m = \Omega(\log(n/\delta))$, then with high probability, if data i and data j are in the same class, we have*

$$card\big(\mathcal{TL}_i(0) \cap \mathcal{TL}_j(0)\big) \approx \frac{\pi - \arccos(\boldsymbol{x}_i^\top \boldsymbol{x}_j)}{4\pi} m.$$

The next important lemma characterizes the evolution of the neuron partition under GD dynamics.

Lemma 6.2 (Informal Lemma A.5)**.** *With high probability, for any $i \in [n]$ and $t \le T$ we have:*
(S1) True-living neurons remain true-living: $\mathcal{TL}_i(t) \subset \mathcal{TL}_i(t+1)$.
(S2) False-dead neurons remain false-dead: $\mathcal{FD}_i(t) \subset \mathcal{FD}_i(t+1)$.
(S3) True-dead neurons remain true-dead or turn true-living: $\mathcal{TD}_i(t) \subset \mathcal{TD}_i(t+1) \cup \mathcal{TL}_i(t+1)$.
***(S4)** False-living neurons turn false-dead in the first step ($t = 0$): $\mathcal{FL}_i(0) \subset \mathcal{FD}_i(1)$.*

To interpret Lemma 6.2, we display the first-step dynamics of four neurons in Figure 1.

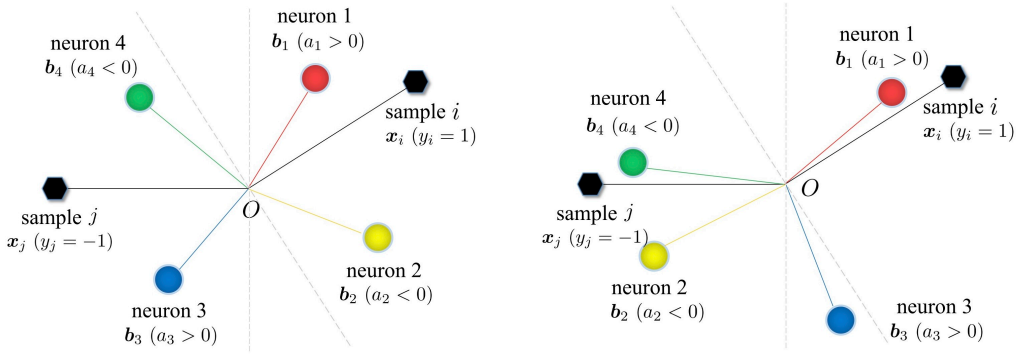
6.1.2 Estimate of hitting time and parameters

Before the analysis of loss descent, we need to estimate the hitting time T and the speed of change for each $a_k, \boldsymbol{b}_k$ in the early stage. Our tool is comparing the hitting time T with an exponentially growing hitting time T_e (see (12) in Appendix A).

$$T_e := \sup\Big\{t \in \mathbb{N}:\ (1+2\eta)^{2(t+1)} - (1-2\eta)^{2(t+1)} \lesssim 1;\ \ (1+2\eta)^{t+1} \lesssim 1\Big\}.$$

Generally speaking, we can prove the following facts.

Lemma 6.3 (Informal Lemma A.8 and A.9)**.** *For any sample $i \in [n]$ and $t \le T_e$, we have:*
(P1) For any $k \in \mathcal{TL}_i(t)$, $a_k(t)sgn(a_k(0))$ is non-decreasing, and $a_k(t)$, $\boldsymbol{b}_k(t)$ have an exponential upper bound.
(P2) For any $k \in [m]$, $a_k(t)$ has an exponential lower bound.
(P3) $f\big(\boldsymbol{x}_i;\boldsymbol{\theta}(t)\big)$ has an exponential upper bound.
***(P4)** $T \ge T_e \ge T^* = \Theta(\frac{1}{\eta})$.*



(a) $t = 0$. (b) $t = 1$.

Figure 1: The first-step dynamics of four neurons which belong to different neuron partitions at $t = 0$ for sample i (with label 1) and sample j (with label -1). Fig 1(a) and Fig 1(b) depict the directions of each neuron at $t = 0$ and $t = 1$, respectively. Results in Lemma 6.2 can be visualized: (S1) neuron $1 \in \mathcal{TL}_i(0)$ and $1 \in \mathcal{TL}_i(1)$. (S2) neuron $4 \in \mathcal{FD}_i(0)$ and $4 \in \mathcal{FD}_i(1)$. (S3) neuron $3 \in \mathcal{TD}_i(0)$ and $3 \in \mathcal{TD}_i(1)$; neuron $2 \in \mathcal{TD}_j(0)$ and $2 \in \mathcal{TL}_j(1)$. (S4) neuron $2 \in \mathcal{FL}_i(0)$ and $2 \in \mathcal{FD}_i(1)$; neuron $3 \in \mathcal{FL}_j(0)$ and $3 \in \mathcal{FD}_j(1)$.

6.1.3 Gram matrix and gradient lower bound

The gradient lower bound is usually an important technical component in the analysis of loss descent. Our analysis establishes a novel lower bound for the gradient for mildly parameterized neural networks based on the neuron partition.

If we define the error for sample i at t as $e_i(t) = f(\boldsymbol{x}_i; \boldsymbol{\theta}(t)) - y_i$, and the Gram matrix at t as $\mathbf{G}(t) = (\nabla f(\boldsymbol{x}_i; \boldsymbol{\theta}(t))^\top \nabla f(\boldsymbol{x}_j; \boldsymbol{\theta}(t)))_{(i,j)\in[n]\times[n]}$. For the analysis of GD, loss descent is determined by $\|\nabla \mathcal{L}(\boldsymbol{\theta}(t))\|^2 = \sum_{i=1}^n \sum_{j=1}^n e_i(t) \mathrm{G}_{i,j}(t) e_j(t)$. Qualitatively, bigger gradient leads to the faster descent of the loss. To derive a tight lower bound for the gradient, we first derive the following bound for the entries of the Gram matrix:

Lemma 6.4 (Informal Lemma A.11)**.** *Under the condition of Theorem 4.2, with high probability, the Gram matrix $\mathbf{G}(t)$ has the following form for any $1 \le t \le T$ and $i, j \in [\frac{n}{2}]$:*

$$\mathbf{G}(t) = \begin{pmatrix} \mathbf{G}_+(t) & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{G}_-(t) \end{pmatrix},$$

$$G_+(t)_{i,j},\ G_-(t)_{i,j} \gtrsim \boldsymbol{x}_i^\top \boldsymbol{x}_j \Big(\frac{\pi - \arccos(\boldsymbol{x}_i^\top \boldsymbol{x}_j)}{4\pi} - \mathcal{O}(\sqrt{\frac{\log(n/\delta)}{m}}) \Big) > 0.$$

Combining the estimate above as well as estimates on parameters and the prediction function, we establish our gradient lower bound.

Lemma 6.5 (Informal Lemma A.12)**.** *For any $1 \le t \le T^*$, we have:*

$$\|\nabla \mathcal{L}(\boldsymbol{\theta}(t))\| = \Omega\Big(1 - (1 + 2\eta)^{2t}\Big).$$

Besides, since we consider GD, the following per-step estimate of loss function (Bubeck et al., 2015) as well as the Hessian upper bound is also useful.

Lemma 6.6 (Informal Lemma A.13 and A.14)**.** *For $0 \le t \le T^* - 1$, we have:*

$$H := \sup_{\boldsymbol{\theta} \text{ along GD trajectory}} \left\|\nabla^2 \mathcal{L}(\boldsymbol{\theta})\right\| = \mathcal{O}(1),$$

$$\mathcal{L}(\boldsymbol{\theta}(t+1)) \le \mathcal{L}(\boldsymbol{\theta}(t)) + \langle \nabla \mathcal{L}(\boldsymbol{\theta}(t)), \boldsymbol{\theta}(t+1) - \boldsymbol{\theta}(t) \rangle + \frac{1}{2} H \left\|\boldsymbol{\theta}(t+1) - \boldsymbol{\theta}(t)\right\|^2.$$

6.2 Proof outline of Theorem 4.2

First, we can estimate the initial neuron partition (Lemma 6.1) at the initialization. Then, we study the dynamics of neuron partition trained by GD (Lemma 6.2), which gives a precise directional characterization about each neuron for each sample. Next, by comparing with an exponentially growing hitting time, we can estimate the hitting time T and the parameters (Lemma 6.3). Combining Lemma 6.1, 6.2 and 6.3, we derive a dynamical lower bound of each element of Gram matrix (Lemma 6.4), which induces our gradient lower bound (Lemma 6.5). Finally, by combining the gradient lower bound (Lemma 6.5) and the loss upper bound (Lemma 6.6), we complete our proof. For more details, please refer to Appendix A.

The proof of Theorem 4.6 is similar to Theorem 4.2 and put into Appendix B.

6.3 Proof outline of Theorem 5.2 and 5.3

The proofs of Theorem 5.2 and 5.3 still rely on the dynamical analysis of neuron partition for each sample. First, the estimate of initial neuron partition (Lemma 6.1) also holds, and the dynamical analysis of neuron partition in Lemma 6.2 holds for any $t \geq 1$ (Lemma C.5 and C.6). Second, we can prove that the structure and lower bound of Gram matrix in Lemma 6.4 hold for any $t \geq 1$ (Lemma C.9), which induces the gradient lower bound below. Finally, combining the estimate of gradient lower bound, Hessian upper bound and loss upper bound, we complete our proof. For more details, please refer to Appendix C and D.

Lemma 6.7 (Informal Lemma C.10)**.** *Under the condition of Theorem 5.3, with high probability, we have the following lower bound for any $t \geq 1$:*

$$\|\nabla \mathcal{L}(\boldsymbol{\theta}(t))\| \gtrsim \mathcal{L}(\boldsymbol{\theta}(t)).$$

In Lyu and Li (2019); Chatterji et al. (2021b), gradient lower bounds are derived for the late stage of training when $\mathcal{L}(\boldsymbol{\theta}(t)) < 1/n$, and take the form $\|\nabla \mathcal{L}(\boldsymbol{\theta}(t))\| \gtrsim \mathcal{L}(\boldsymbol{\theta}(t)) \log\big(1/\mathcal{L}(\boldsymbol{\theta}(t))\big)/\|\boldsymbol{\theta}(t)\|$. Our lower bound, instead, works on the whole training process. With this stronger gradient lower bound, we can show the exponential convergence in Theorem 5.3.

7 Conclusion and Future Work

In this paper, we study the convergence of GD and SGD when training mildly parameterized neural networks. On one hand, we show early stage convergence for a wide range of loss functions, optimization algorithms, and training data distributions. On the other hand, under some assumptions on the loss function and data distribution, we show global convergence of GD. Our analysis can be extended to the minimization of population risk (See Appendix E).

The theoretical understanding of the training of neural networks, especially for neural networks with practical sizes, still has a long way to go. For instance, although the late stage training for exponential loss function is simple and clear, that for other losses is much more complicated. Phenomena like unstable convergence (Ahn et al., 2022; Wu et al., 2018; Cohen et al., 2021) happen. Better understanding of these phenomena during training is an important direction of future work.

References

Kwangjun Ahn, Jingzhao Zhang, and Suvrit Sra. Understanding the unstable convergence of gradient descent. *arXiv preprint arXiv:2204.01050*, 2022.

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 2019a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019b.

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

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019b.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Niladri S Chatterji, Philip M Long, and Peter Bartlett. When does gradient descent with logistic loss interpolate using deep networks with smoothed relu activations? In *Conference on Learning Theory*, pages 927–1027. PMLR, 2021a.

Niladri S Chatterji, Philip M Long, and Peter L Bartlett. When does gradient descent with logistic loss find interpolating two-layer networks? *Journal of Machine Learning Research*, 22(159):1–48, 2021b.

Patrick Cheridito, Arnulf Jentzen, Adrian Riekert, and Florian Rossmannek. A proof of convergence for gradient descent in the training of artificial neural networks for constant target functions. *Journal of Complexity*, page 101646, 2022.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

Amit Daniely. Sgd learns the conjugate kernel class of the network. *Advances in Neural Information Processing Systems*, 30, 2017.

Tian Ding, Dawei Li, and Ruoyu Sun. Sub-optimal local minima exist for neural networks with almost all non-linear activations. *arXiv preprint arXiv:1911.01413*, 2019.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

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

Weinan E, Chao Ma, Qingcan Wang, and Lei Wu. Analysis of the gradient descent algorithm for a deep neural network model with skip-connections. *arXiv preprint arXiv:1904.05263*, 2019.

Weinan E, Chao Ma, and Lei Wu. A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics. *Science China Mathematics*, 63(7):1235, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Arnulf Jentzen and Adrian Riekert. A proof of convergence for the gradient descent optimization method with random initializations in the training of neural networks with relu activation for piecewise linear target functions. *arXiv preprint arXiv:2108.04620*, 2021.

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33:17176–17186, 2020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in Neural Information Processing Systems*, 31, 2018.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30, 2017.

Yuanzhi Li, Tengyu Ma, and Hongyang R Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In *Conference on learning theory*, pages 2613–2682. PMLR, 2020.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. *Advances in neural information processing systems*, 27, 2014.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. *arXiv preprint arXiv:1906.05890*, 2019.

Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. *Advances in Neural Information Processing Systems*, 34, 2021.

Chao Ma, Lei Wu, and Weinan E. The quenching-activation behavior of the gradient descent dynamics for two-layer neural network models. *arXiv preprint arXiv:2006.14450*, 2020.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory*, pages 2388–2464. PMLR, 2019.

Mary Phuong and Christoph H Lampert. The inductive bias of relu networks on orthogonally separable data. In *International Conference on Learning Representations*, 2020.

Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. In *International conference on machine learning*, pages 4433–4441. PMLR, 2018.

Itay M Safran, Gilad Yehudai, and Ohad Shamir. The effects of mild over-parameterization on the optimization landscape of shallow relu neural networks. In *Conference on Learning Theory*, pages 3889–3934. PMLR, 2021.

Ruoyu Sun, Dawei Li, Shiyu Liang, Tian Ding, and Rayadurgam Srikant. The global landscape of neural networks: An overview. *IEEE Signal Processing Magazine*, 37(5):95–108, 2020.

Yuandong Tian. 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*, pages 3404–3413. PMLR, 2017.

Lei Wu. Learning a single neuron for non-monotonic activation functions. *arXiv preprint arXiv:2202.08064*, 2022.

Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31: 8279–8288, 2018.

Gilad Yehudai and Shamir Ohad. Learning a single neuron with gradient methods. In *Conference on Learning Theory*, pages 3756–3786. PMLR, 2020.

Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization (2016). *arXiv preprint arXiv:1611.03530*, 2017.

Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International conference on machine learning*, pages 4140–4149. PMLR, 2017.

Mo Zhou, Rong Ge, and Chi Jin. A local convergence theory for mildly over-parameterized two-layer neural network. In *Conference on Learning Theory*, pages 4577–4632. PMLR, 2021.

Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The global convergence results in Section 5 are built on restrictive assumptions of the loss function and training data.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] In Section 3, 4 and 5.
   (b) Did you include complete proofs of all theoretical results? [Yes] In Appendix A, B, C, D, F and G.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]