When Expressivity Meets Trainability: Fewer than n Neurons Can Work

Anonymous Author(s)
Affiliation
Address
email

Abstract

Training a narrow neural network remains challenging. One major difficulty lies in its highly wrinkled empirical loss landscape, which jeopardizes the 'trainability'; another issue is its possibly limited representation power, a.k.a. the 'expressivity'. We ask two questions: Do narrow networks have as strong expressivity as wide ones? If so, how to find such a solution via optimization? In this work, we provide partially affirmative answers to both questions for 1-hidden-layer networks with fewer than n (sample size) neurons. We show that: as long as the width satisfies $m \geq \frac{2n}{d}$ (where d is the input dimension), then its expressivity is strong, i.e., there exists at least one global-min with zero training loss with probability one. In addition, it is surrounded by good landscape with no local-min or saddles. To find such a global-min, we propose a training regime based on a specially designed constrained formulation, in which first-order optimality condition implies the near-global optimality for the unconstrained training problem, so the trainability is guaranteed. Our proposed training regime, where only one or two lines of code change are involved, significantly improves the performance of narrow networks in our experiments.

1 Introduction

Learning a neural network requires analyzing its 'trainability' (how effective a network can be optimized) and 'expressivity' (how complex the function a network can represent) [1]. As for trainability, it is widely accepted that large width is one critical factor that contributes to successful training. One evidence is that increasing the width of neural networks helps create benign empirical loss landscape ([2–4]), while narrow networks (width $m <$ sample size n) suffer from highly non-convex landscape with numerous bad local minima ([5–9]). Another evidence is that pruned models can achieve similar performance to the original model (e.g. [10]), which implies that many redundant parameters only help optimization, but have little use for inference. In this sense, it is possible to directly learn a powerful narrow network without the extra pruning procedure, but one has to bypass the hurdle of non-convexity. Nevertheless, reducing the size of neural networks is appealing in terms of saving heavy computational burdens and large power consumption. Further, a smaller network is more suitable for embedded systems and small portable devices.

Let us first describe the typical analysis framework for general neural network training. Formally, given the dataset $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$, we consider the 1-hidden-layer neural networks in the general form:

$$f(x;\theta) = \sum_{j=1}^{m} v_j \sigma\left(w_j^T x\right), \tag{1}$$

[1] The terms 'expressivity' and 'trainability' are inspired by Chen [1].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

where $\sigma\left(w_j^T x\right)$ is the output of the j-th hidden nodes with hidden weights w_j, $\sigma(\cdot)$ is the activation function, and v_j is the corresponding outer weight (bias terms are ignored for simplicity). To learn such a neural network, we search for the optimal parameter $\theta = (w, v)$ by minimizing the following (training) loss:

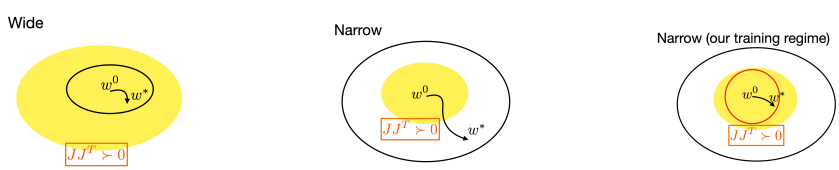
$$\min_{\theta} \ell(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - f\left(x_i; \theta\right)\right)^2, \tag{2}$$

The gradient of the above problem w.r.t. hidden weights $w = \{w_i^{d\times 1}\}_{i=1}^m$ is given by: $\nabla_w \ell(\theta) = J(w;v)^T (f(w;v) - y) \in \mathbb{R}^{md\times 1}$, where $J(w;v) \in \mathbb{R}^{n\times md}$ is the Jacobian matrix w.r.t w:

$$J(w;v) := \begin{bmatrix} \nabla_w f\left(w; x_1, v\right)^T \\ \vdots \\ \nabla_w f\left(w; x_n, v\right)^T \end{bmatrix} = \begin{pmatrix} v_1 \sigma'\left(w_1^T x_1^T\right) x_1^T & \cdots & v_m \sigma'\left(w_m^T x_1\right) x_1^T \\ & \vdots & \\ v_1 \sigma'\left(w_1^T x_n\right) x_n^T & \cdots & v_m \sigma'\left(w_m^T x_n\right) x_n^T \end{pmatrix} \in \mathbb{R}^{n\times md}.$$

When first order methods like (stochastic) gradient descent (GD & SGD) are used, a solution $\theta^* = (w^*, v^*)$ with nearly 0 gradient is returned [11]. For problem (2), this means that if $J(w;v) \in \mathbb{R}^{n\times md}$ is full row rank (when $n < md$), then GD directly returns a global-min with zero training loss. Fortunately, recent works have shown that indeed in many cases, it is possible to guarantee the full rankness of the Jacobian matrix along the training trajectories because of the following arguments: (i) with high probability, $J(w;v)$ is non-singular locally around the random initialization ([12], [13]), (ii) increasing the width can effectively bound the parameter movement from initialization, so these "nice" properties would maintain throughout the training, leading to a linear convergence rate [14]. Under this general framework, there are many state-of-art convergence results for wide networks with width $\Omega(\text{poly}(n))$, including the Neural Tangent Kernel (NTK) regime for the infinite width ([15–22]). This idea is also illustrated in Figure 1, (a).

However, the above arguments no longer hold when the width is not large enough to control the movement of hidden weights. In this case, the iterates may easily travel far away from the initial point and get stuck at some singular-Jacobian points with high training loss (see Figure 1, (b)). This argument is supported by our experiment in Section 4. At this point, it is unknown if the narrow network can be easily trained to achieve zero loss.



(a) Wide networks (b) Narrow networks (c) Narrow networks (our regime)

Figure 1: The parameter movement under different regime. the shaded area indicates the region where $J(w;v)$ is non-singular, the black circle denotes the region that GD iterates will explore, and the red circle is the constraint designed in our training regime.

Further, the expressivity (a.k.a. the representation power) of narrow networks is also difficult to analyze. Expressivity is not an issue for wide networks with width $m \geq n$, since the feature matrix

$$\Phi(w) := \begin{bmatrix} \sigma\left(w_1^T x_1\right), \ldots, \sigma\left(w_m^T x_1\right) \\ \vdots \\ \sigma\left(w_1^T x_n\right), \ldots, \sigma\left(w_m^T x_n\right) \end{bmatrix} \in \mathbb{R}^{n\times m}$$

can span the whole $\mathbb{R}^n$ given it is full row rank, so we can perfectly match any label $y \in \mathbb{R}^n$ even without tuning the hidden layer. This idea is used in the random feature regime [23]. However, when the width fails to exceed the sample size, the expressivity of the network is unclear since the rank of the feature matrix is strictly smaller than n. In other words, it is not clear whether an global-min with zero loss exists.

The above discussion can be summarized using the following two questions:

(Q1) *Can a narrow network have as strong* **expressivity** *as a wide one?*

(Q2) *If so, is the network* **trainable** *so that gradient methods can find such a solution?*

The key challenges in answering these questions are listed below:

- It is not clear whether a solution with strong expressivity *exists* for narrow networks. Many existing works focus on verifying the relationship between expressive global solutions and stationary points, but they neglect the (non)existence of such solutions (e.g. [12], [13]).
- Even if such expressive solutions do exist, it is still not clear how to reach those solutions because the landscape of a narrow neural network can be highly non-convex.
- Even assuming that we can identify a region that contains expressive solutions and has a good landscape, it is potentially difficult to keep the iterates inside such a good region. One may think of imposing an explicit constraint, but this approach might introduce bad local-mins on the boundary [11].

In this work, we (partially) answer **(Q1)** and **(Q2)** for a 1-hidden-layer networks with fewer than n neurons. Our main contributions are as follows:

- We prove that, as long as the width is larger than $m \geq \frac{2n}{d}$, then the expressivity of the network is strong, i.e., w.p.1. there exists at least one global-min with zero loss. In addition, it is surrounded by a good landscape with no local-min or saddles.
- To find such a global-min, we propose a training regime in which first-order optimality condition implies the near-global optimality. To be more specific, we propose a constrained optimization problem that restricts the hidden weights to be close to the identified nice region. We show that every Karush–Kuhn–Tucker (KKT) point is an approximated global-min of the unconstrained training problem, that is, no additional local-mins are introduced in our formulation.
- In real-data experiment, our proposed training regime can significantly improve the performance of narrow networks.

2 Related works

In addition to the aforementioned works, one of the most related works is by Soudry and Carmon [13]. They show that, for 1-hidden-layer networks with piecewise linear activation function, if $md > n$, then every differentiable local-min has zero loss. However, in their analysis, the full-rankness of Jacobian is ensured by extra conditions such as weight perturbation, and it is unclear whether such extra conditions still hold in practice. Given $md > n$, Xie et al. [12] also formulates a Polyak-Lojasiewicz (PL) condition of 1-hidden-layer neural networks if the Jacobian is full rank. But again, their full-rankness relies on an extra condition called weight discrepancy, which may not hold after the training procedure. Additionally, these two works did not discuss whether a stationary solution exists around the initial point.

The expressivity of neural networks has been a popular topic for decades (e.g. [24]). Particularly, many such works show that certain functions can be efficiently expressible by neural networks with a small number of parameters ([25–29]), but such a network has to be deep enough. Additionally, their discussions do not involve trainability. Shalev et al. [30] pointed out that many expressible functions are not efficiently learnable, at least by gradient descent, so it is unclear whether the global minimum can be found in practice.

One recent work by Zhou et al. [31] discusses the expressivity and trainability together by studying the local convergence theory of mildly over-parameterized 1-hidden-layer networks. They show that, any stationary point with a small enough loss is a global minimum. They further propose an initialization strategy that provably returns a small loss under a mild assumption on the width. However, their proposed initialization is costly to find since it requires solving an additional optimization problem.

3 Main results

3.1 Problem Settings and Preliminaries

We denote $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$ as the training samples, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$. For theoretical analysis, we focus on 1-hidden-layer neural networks $f(x;\theta) = \sum_{j=1}^m v_j \sigma\left(w_j^T x\right) \in \mathbb{R}$, where $\sigma(\cdot)$ is the activation function, $w_j \in \mathbb{R}^d$ and $v_j \in \mathbb{R}$ are the parameters to be trained. Note that we only consider the case where $f(x;\theta) \in \mathbb{R}$ has 1-dimensional output for notation simplicity. To learn such a neural network, we search for the optimal parameter $\theta = (w, v)$ by

minimizing the empirical loss (2), and sometimes we also use $\ell(w;v)$ or $f(w;x,v)$ to emphasize the role of w. We use the following shorthanded notations: $x := (x_1^T;\ldots;x_n^T) \in \mathbb{R}^{n\times d}$, $y := (y_1,\ldots,y_n)^T \in \mathbb{R}^n$, $w := (w_1,\ldots,w_m)_{j=1}^m \in \mathbb{R}^{d\times m}$, $v := (v_1,\ldots,v_m)_{j=1}^m \in \mathbb{R}^m$, and $f(w;v) := (f(x_1;w,v), f(x_2;w,v),\ldots,f(x_n;w,v))^T \in \mathbb{R}^n$. We denote the Jacobian matrix of $f(w;v)$ w.r.t w as

$$J(w;v) := \begin{bmatrix} \nabla_w f(w;x_1,v)^T \\ \vdots \\ \nabla_w f(w;x_n,v)^T \end{bmatrix} = \begin{pmatrix} v_1\sigma'\left(w_1^T x_1\right)x_1^T & \cdots & v_m\sigma'\left(w_m^T x_1\right)x_1^T \\ & \vdots & \\ v_1\sigma'\left(w_1^T x_n\right)x_n^T & \cdots & v_m\sigma'\left(w_m^T x_n\right)x_n^T \end{pmatrix} \in \mathbb{R}^{n\times md}.$$

We define the feature matrix

$$\Phi(w) := \begin{bmatrix} \sigma\left(w_1^T x_1\right),\ldots,\sigma\left(w_m^T x_1\right) \\ \vdots \\ \sigma\left(w_1^T x_n\right),\ldots,\sigma\left(w_m^T x_n\right) \end{bmatrix} \in \mathbb{R}^{n\times m}.$$

Throughout the paper, 'w.p.1' is the abbreviation for 'with probability one'; when we say 'in the neighborhood of initialization', it means 'w is in the neighborhood of the initialization w^0'.

Next, let us describe the *mirrored LeCun's* initialization in Algorithm 1, which is the key element in our analysis. The idea is that through this initialization, the hidden outputs will cancel out with the outer weights, so that we get 0 initial output for any input x; see Fig. 2 for a simple illustration. This initialization is crucial in our analysis, and more intuition will be provided shortly.

Algorithm 1 The mirrored LeCun's initialization

1: Initialize all the weights using LeCun's initialization: $w_{i,j}^0 \sim N(0,\frac{1}{d})$, $v_i^0 \sim N(0,\frac{1}{m})$, for $i=1,\ldots,m$, $j=1,\ldots,d$.
2: Reset $(w_{\frac{m}{2}+1}^0,\ldots,w_m^0) \leftarrow (w_1^0,\ldots,w_{\frac{m}{2}}^0)$, and set $(v_{\frac{m}{2}+1}^0,\ldots,v_m^0) \leftarrow (-v_1^0,\ldots,-v_{\frac{m}{2}}^0)$

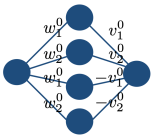


Figure 2: A simple example of the mirrored LeCun's initialization.

Throughout the paper, we will make the following assumptions.

Assumption 1. *For $f(x;\theta)$ in (1), we assume its width m is an even number, and $m \geq \frac{2n}{d}$.*

Assumption 2. *We assume the activation function $\sigma(\cdot):\mathbb{R}\to\mathbb{R}$ is analytic and L-lipschitz continuous, its zero set only contains 0: $\{z|\sigma(z)=0\}=\{0\}$, and its first n derivatives at 0, i.e., $\sigma(0)$, $\sigma'(0),\cdots,\sigma^{(n-1)}(0)$, are all non-zero.*

Assumption 3. *$x_1,\cdots,x_n$ are independently sampled from a continuous distribution in $\mathbb{R}^d$.*

When $d>2$, Assumption 1 can be applied to narrow networks with $m<n$. Assumption 2 covers many commonly used activation functions such as sigmoid, softplus, and Tanh, but it does not cover ReLU since it it nonsmooth.

3.2 Expressivity analysis

In this section, we prove the strong expressivity of neural networks under the previous assumptions. Further, such an expressive solution is surrounded by a good landscape with no local-min or saddles.

Theorem 1. *Suppose Assumption 1, 2, and 3 holds. If the neural network $f(x;\theta) = \sum_{j=1}^m v_j\sigma\left(w_j^T x\right)$ is initialized at the mirrored LeCun's initialization given in Algorithm 1, with $\theta^0=(w^0,v^0)$, then for any small neighborhood around w^0: $B_\epsilon(w^0)=\{w \mid \|w-w^0\|_F \leq \epsilon\}$, there exists a $w\in B_\epsilon(w^0)$ and a entry-wise non-zero v, such that, w.p.1., for all $\{(x_i,y_i)\}_{i=1}^n$, the output of f will be exactly the groundtruth label:*

$$f(x_i;\theta) = \sum_{j=1}^m v_j\sigma(w_j^T x_i) = y_i, i=1,\cdots,n. \tag{3}$$

In addition, every stationary point $\theta^*=(w^*,v^*)$ is a global-min if it satisfies: $\nabla_w\ell(w^*;v^*)=0$, $w^*\in B_\epsilon(w^0)$ and entry-wise non-zero v^*.

Remark 1. *When $d > 2$, Theorem 1 is applicable to narrow networks with $m < n$. Despite the small number of neurons in the hidden layer, Theorem 1 claims a strong expressivity, i.e., there exists at least one global-min with zero loss w.p.1., and it is surrounded by a good landscape with no local-min or saddles. Especially, the benign landscape suggests that such a global solution is attainable by the first-order algorithms, which motivates our trainability analysis in the following section.*

Remark 2. *Theorem 1 also emphasizes the role of hidden weights of a neural network: it is a key ingredient for strong expressivity. When $m < n$, if we fix all the $w = w^0$, the range space of the feature matrix $\Phi(w^0)$ does not cover the whole $\mathbb{R}^n$ space, so there always exists a label y, such that no v^* can be found that perfectly maps the input to y. However, a small tolerance of the movement of w will let $f(x;\theta)$ perfectly fit any input-label pair, so the movement of w is vitally important. The free perturbation of w serves as an effective remedy against the limited expressivity.*

Proof sketch. Theorem 1 consists of two arguments: (i) there exists a global-min with zero loss, (ii) in the neighborhood of initialization, every stationary point is a global-min. A detailed proof is relegated to the supplemental material. We outline the main idea below.

To argue (i), the key idea is to make use of the Inverse (Implicit) Function Theorem (IFT), which is stated in the supplemental material. According to IFT, as long as a $n \times n$ submatrix of $J(w^0; v^0)$ is invertible, then for any $y \in \mathbb{R}^n$, there exists a $w^* \in B_\epsilon(w^0)$ whose prediction output $f(w^*; v^0) \propto y - f(w^0; v^0)$. Additionally, since $f(w^0; v^0) = 0$, we have $f(w^*; v^0) \propto y$. Once this is shown, then we just need to scale all the outer weight v_j uniformly and the output will be exactly y since $f(w^*; v)$ is linear in v. The detailed proof can be seen in the supplemental material.

To argue (ii), recall that all the stationary points $\theta^* = (w^*, v^*)$ satisfy $\nabla_w \ell(\theta^*) = J(w^*; v^*)^T (f(w^*; v^*) - y) = 0$. Therefore, if $[J^T(w^*; v^*)] \in \mathbb{R}^{md \times n}$ is of full column rank, the stationary point $\theta^* = (w^*, v^*)$ is a global minimizer with $\ell(\theta^*) = 0$. The desired full-rankness condition is true because: (i) $J(w^0; v^0)$ is full rank w.p.1 at initialization, (ii) w^* will not leave the small neighborhood $B_\epsilon(w^0)$, so the dynamics of w stays inside the manifold of full-rank Jacobian.

□

The proof of Theorem 1 relies on the full-rankness of the Jacobian matrix. We note that such full-rankness holds for *both* the mirrored and the regular LeCun's initialization (the case for the regular one can be proved using the same technique). So why do we insist on shifting the initial output to 0? Simply put, the 'randomness of weights' contributes to the 'full-rankness', while the 'shifting' allows us to gain representation power by applying IFT properly. To be more specific, $f(w^0, v^0) = 0 \in \mathbb{R}^n$ is important because it is surrounded by all possible directions pointed from $0 \in \mathbb{R}^n$, so for any $y \in \mathbb{R}^n$, IFT claims that there exists at least one $w^* \in B_\epsilon(w^0)$, s.t. $f(w^*; v^0) \propto y - f(w^0; v^0) = y$, therefore, $f(w^*; v^*)$ can perfectly match y by scaling v^0 by some constant (Figure 3(a) illustrates this case when $n = 2$).

In contrast, if we use IFT around regular LeCun's initialization, the existence of $f(w^0; v^0)$ on the right hand side resists us from scaling v^0 like before (Figure 3(b) illustrates this case). A natural question is: Do narrow networks also have strong expressivity around the *regular* LeCun's initialization? At this point, we do not have enough theoretical support to answer this question. However, our experiments in Section 4 provide a negative answer: training will fail if we only search around the regular LeCun's initialization.

(a) The mirrored LeCun's initialization (b) Regular LeCun's initialization

Figure 3: Examples of using Inverse Function Theorem (IFT) under different initialization strategies. The illustration here is for $n = 2$. For (a), scaling v^0 will directly lead to zero loss, while it is not true for (b) due to the non-zero $f(w^0; v^0)$.

The idea of 0 initial output is also used in other recent works in Table 1. For instance, Lee et al. [20] and Chizat et al. [22] use 0 initial output in the NTK regime to eliminate the bias term and simplify their proof. Nguyen [19] also uses 0 initial output, but their initialization is very different in that all the hidden layers will have high values while the last layer is assigned to 0. In this way, they manage to limit the movement of hidden layers without increasing the width. Unlike all these works, we utilize the 0 initial output to identify the strong expressivity of a narrow neural network. To our best knowledge, this is the first time that the 0 initial output property has been linked to expressivity.

Table 1: Comparison of recent works considering zero initial output

Work	Width	Motivation
Lee et al [20]	$m_{L-1} \to \infty$	To avoid handling the bias term in the NTK regime
Chizat et al [22]	$m_{L-1} \to \infty$	To avoid handling the bias term in the NTK regime
Nguyen et al [19]	$m_{L-1} = \Omega(n)$	To ensure linear convergence via imbalanced weight
Ours	$m = \Omega(\frac{n}{d})$	To earn the expressivity via the Inverse Function Theorem

3.3 Trainability analysis

Despite the expressivity and good local properties stated in Theorem 1, in practical training, the weights can easily escape the nice neighborhood, especially when the width is not sufficiently large. To keep the hidden weights inside this nice region, an intuitive idea is to impose an explicit constraint, but it may suffer from bad local-min on the boundary with a very large loss [11]. This view is confirmed by our experiments in Section 4, Figure 5, (b): when we add the hidden-weight constraint directly to the regular training regime (i.e., let $\|w - w^0\|_F \leq \epsilon$), it fails to find a low-cost solution when ϵ and width become small.

In this sense, we need to design a constrained problem, such that all the KKT points will have small training loss, including those on the boundary. Fortunately, Theorem 1 suggests one such formulation. Recall in the proof of Theorem 1, we construct a 0-loss global-min by scaling v^0 , so v^* still follows the pairwise-opposite pattern. Inspired by this, we consider the following neural network [we abuse the notation of $f(x;\theta)$, $f(x;w,v)$ and $v = (v_1, \ldots, v_{\frac{m}{2}})$ here]:

$$f(x;w,v) = \sum_{j=1}^{\frac{m}{2}} v_j \left(\sigma(w_j^T x) - \sigma(w_{j+\frac{m}{2}}^T x)\right). \tag{4}$$

Note that the optimization variable for the outer layer is only $v = (v_1, \ldots, v_{\frac{m}{2}})$, and the rest of the outer weights are automatically set to be $-v$. Despite the change of v, Theorem 1 still applies since it does not have any specific requirement on v. We then use (4) to formulate the following constraint problem (5):

$$\min_{\theta} \ell(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i;\theta))^2, \quad \text{s.t.} \qquad w \in B_\epsilon(w^0),\ v \in B(v) \tag{5}$$

where

$$B_\epsilon(w^0) := \{w \mid \|w - w^0\|_F \leq \epsilon\}$$
$$B(v) := \{v \mid \exists\, \zeta > 0,\ \kappa < \infty \text{ s.t. } v \geq \zeta \mathbf{1},\ \text{and } v_j/v_{j'} \leq \kappa, \forall\, (j,j') \in \{1, \cdots, m\}\}.$$

Here, $\zeta > 0$ is a small constant that keeps v to be entry-wise non-zero, which is an essential requirement of Theorem 1. The requirement of $v_j/v_{j'} \leq \kappa < \infty$ allows all entries of v to be uniformly large, but it rules out the case when some entries are increasing much faster than others. Instead of regarding all these requirements of $B(v)$ as prior assumptions, we formulate them into the constraints in the problem, as a result, all the iterates in the practical training algorithm will strictly follow these requirements. In Theorem 2, we show that every KKT point of problem (5) implies the near-global optimality for the unconstrained training problem (2).

Theorem 2. *Under Assumption 1, 2, and 3, assume the neural network $f(x;\theta) = \sum_{j=1}^{m} v_j \sigma\left(w_j^T x\right)$ is initialized at the mirrored LeCun's initialization given in Algorithm 1, with $\theta^0 = (w^0, v^0)$, then every KKT point $\theta^* = (w^*, v^*)$ of (5) will be an approximated global-min w.p.1., that is:*

$$\ell(w^*, v^*) = O(\epsilon^2). \tag{6}$$

Proof sketch. The proof is built on the special structure of neural network $f(x;\theta)$, including the linear dependence of v and the mirrored pattern of parameters. Here, we describe our high level idea and the analysis roadmap. The proof consists of proving the following claims:

(I) every KKT point θ^* satisfies $\|\nabla_w \ell(w^*;v^*)\| = O(\epsilon)$;

(II) the gradient of w always dominates the error term, i.e. $\|\nabla_w \ell(w;v)\|_2^2 = \Omega\left(\ell(w^*;v^*)\right)$, so we have $\ell(w^*;v^*) = O(\epsilon^2)$.

To prove claim (I), we only need to consider the case where w^* is on the boundary and $\|\nabla_w \ell(w^*;v^*)\|_2 \neq 0$ (otherwise Theorem 2 automatically holds based on Theorem 1). In this case, by the optimality condition, taking $\eta \in \mathbb{R}$ as a small step size, we have

$$-\eta \nabla_w \ell(w^*;v^*) = \eta \frac{\|\nabla_w \ell(w^*;v^*)\|}{\|w^* - w^0\|}(w^* - w^0) = \tilde{\eta}(w^* - w^0), \tag{7}$$

where $\tilde{\eta} = \eta \frac{\|\nabla_w \ell(w^*;v^*)\|}{\|w^* - w^0\|}$. Now, our key observation is that, after moving along (7) from w^*, the change of the loss is not significant due to the special local structure of $f(w;v^*)$, therefore, $\|\nabla_w \ell(w^*;v^*)\|_2$ can be bounded. To be more specific, we denote f^* as the neural network output at the KKT point θ^*; denote $\bar{f}$ as the neural network output after taking a small step η along the negative partial gradient direction of w; and denote f' as an rough estimate of $\bar{f}$:

$$f^* := f\left(w^*;v^*\right) = J(w^0;v^*)w^* + R^*, \tag{8}$$

$$\bar{f} := f\left(w^* - \eta \nabla_w \ell(w^*;v^*);v^*\right) = (1+\tilde{\eta})f^* - (1+\tilde{\eta})R^* + \bar{R}, \tag{9}$$

$$f' := \bar{f}_{\text{lin}} + (1+\tilde{\eta})R^* = (1+\tilde{\eta})f^*, \tag{10}$$

where $\bar{f}_{\text{lin}}$ is the first-order Taylor approximation of $\bar{f}$, R^* is the second-order Taylor residue of f^* (similarly for $\bar{R}$). Additionally, (8), (9), and (10) are due to the the symmetric property of w^0, v^0 and v^*, so the bias terms in the Taylor expansion will vanish. Now, we compare the value of the loss on each of f^*, $\bar{f}$ and f'. Define $\ell \circ f = \frac{1}{2}\|y - f\|_2^2$, we prove the following crucial relationship:

$$\eta \|\nabla_w \ell(w^*;v^*)\|_2^2 \overset{(a)}{\leq} \ell \circ f^* - \ell \circ \bar{f} \overset{(b)}{\leq} \ell \circ f' - \ell \circ \bar{f} = \frac{1}{2}\left\|f' - \bar{f}\right\|_2 \left\|f' + \bar{f} - 2y\right\|_2 \overset{(c)}{=} O(\tilde{\eta}\epsilon^2) \tag{11}$$

Eq. (11) plays a key role in our analysis. Here, (a) can be easily shown by applying Descent lemma in this local region, yet (b) and (c) are not that obvious. Recall in (8), (9), and (10), we know that: (i) although the location of $\bar{f}$ is unclear, f' points at the same direction as f^*. (ii) As an estimator of $\bar{f}$, f' is not far away from it, i.e. $\|\bar{f} - f'\|_2 = \|\bar{R} - (1+\tilde{\eta})R^*\|_2$ only involves the second-order Taylor residue terms. With this observation, (b) is proved in Lemma 1 (stated below) by geometric properties, and (c) is calculated in Lemma 2 (stated below), so the relationship (11) is proved. Therefore, we have θ^* satisfies $\|\nabla_w \ell(w^*;v^*)\| = O(\epsilon)$ by plugging in $\eta = \frac{\epsilon\tilde{\eta}}{\|\nabla_w \ell(w^*;v^*)\|_2}$, and claim (I) is proved.

Lemma 1. *Under the background of Theorem 2, we have $\ell \circ f' \geq \ell \circ f^*$, i.e., $\|f' - y\|_2^2 \geq \|f^* - y\|_2^2$.*

Lemma 2. *Under the background of Theorem 2, we have: $\left\|f' - \bar{f}\right\|_2 \left\|f' + \bar{f} - 2y\right\|_2 = O(\epsilon^2)$.*

As for claim (II), it is true as long as $J(w;v)$ is of full row rank, which has been shown in Theorem 1. A more detailed proof of Lemma 1 and 2, as well as the proof of the whole Theorem 2 are in the supplemental material.

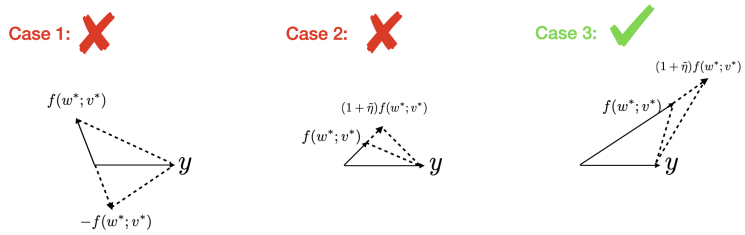


Figure 4: Geometrical illustration of three cases in Lemma 1: comparing with f^*, $f' = (1+\tilde{\eta})f^*$ will not reduce further the distance to y.

Proof sketch of Lemma 1. Here, we provide a proof sketch of Lemma 1, we need to discuss the following cases:

(1) When $f^{*T}y \geq 0$: we prove by contradiction. Since f^* is linear in v^*, $\|(1+\tilde{\eta})f^* - y\|_2^2 < \|f^* - y\|_2^2$ implies $\|f(w^*; (1+\tilde{\eta})v^*) - y\|_2^2 < \|f^* - y\|_2^2$, which means we can further reduce the loss by magnifying v^* to $(1+\tilde{\eta})v^*$, which is still feasible, we have a contradiction to the assumption that $(w,^* v^*)$ is a KKT point (see Figure 4, Middle).

(2) When $f^{*T}y < 0$: changing v^* to $-v^*$ will further reduce the distance to y (see Figure 4, Left), this is a contradiction to the fact that v^* is a KKT point, so case (2) will not happen.

In conclusion, we always have $\ell \circ f' \geq \ell \circ f^*$ (see Figure 4, Right). □

3.4 Discussion: Extension to deep networks

In the previous sections, we analyze the trainability and expressivity of 1-hidden-layer networks with $m \geq \frac{2n}{d}$. To extend our results to L-layer deep networks, we apply the mirrored LeCun's initialization and the constrained formulation (5) to the last two layers and treat the output of the $(L-2)$-th layer as the input features. By Theorem 1, the expressivity is guaranteed if the input follows a continuous distribution. Fortunately, for deep neural networks with $m_l \geq \frac{2n}{m_{l-1}}$ (where m_l is the width of the l-th layer), the output of $(L-2)$-th layer still follows a continuous distribution under Assumption 2 and 3. It can be shown by applying IFT recursively and using a similar technique as Theorem 1. This result is formally stated in the following Lemma 3.

Lemma 3. *Given a deep fully-connected neural network with L layers:*

$$f(x;\theta) = w^{(L)}\sigma\left(w^{(L-1)}\ldots\sigma\left(w^{(2)}\sigma\left(w^{(1)}x\right)\right)\right),$$

where $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ is the activation function, $w^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}}, l = 1, \ldots, L$. Under Assumption 2 and 3, suppose $m_l \geq \frac{2n}{m_{l-1}}$, for $l = 1 \cdots, L-1$, then, for $\{x_1, \cdots, x_n\}$, the outputs of $(L-2)$-th layer follows a continuous distribution.

Proof of Lemma 3 can be seen in the supplemental material. As for the trainability for deep networks, the difficulty lies in the free movement of the input features (which is fixed in the shallow case), it will be considered in future works. Nevertheless, our idea motivates a better training regime for deep networks, and it is numerically verified in our experiments.

4 Experiments

In this section, we provide empirical validation for our theory. Specifically, we compare the performance of two training regimes:

(1) Our training regime: constrained problem (5) with the mirrored LeCun's initialization.

(2) Regular training regime: unconstrained GD-based method with LeCun's initialization.

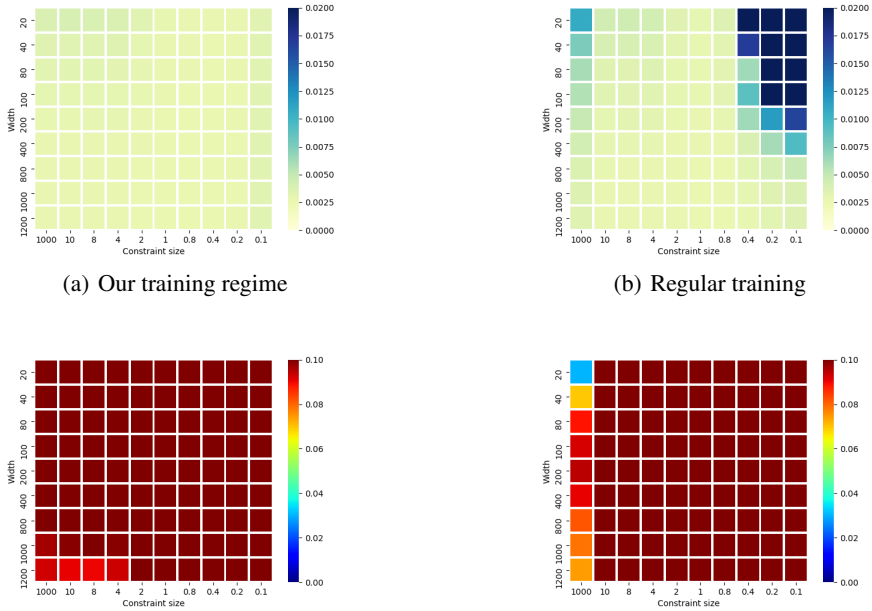
To reach the KKT point of the constrained problem (5), Projected Gradient Descent (PGD) is used (see Bertsekas [11]), and detailed experimental setup are explained in the supplemental material. Here, we only demonstrate the results on the synthetic data and (Restricted) ImageNet [32], and experiments on MNIST [33], CIFAR-10 [34], and CIFAR-100 [34] are given in the supplemental material.

4.1 Results on the synthetic dataset

To begin with, on the synthetic regression dataset, we train 1-hidden-layer networks under different width and different hidden-weight constraint size ϵ in problem (5). Results can be seen in Figure 5, (a) & (b). First, our training regime performs well regardless of width, yet regular unconstrained training fails when the network is narrow (in Figure 5, $\epsilon = 1,000$ corresponds to the unconstrained case). Second, it does not work when we directly impose the hidden-weight constraint of (5) to the regular training regime (and PGD is used accordingly), as it fails to find a low-cost solution when ϵ and width become small. In contrast, our training regime always finds a low-cost solution with any choice of ϵ.

Furthermore, Figure 5, (c) & (d) depict $\lambda_{\min}(J(w^*; v^*))$, i.e. the minimum singular value of the Jacobian at the stationary (or KKT) points. As expected in Section 1, when the width is small, regular training regime (when $\epsilon = 1,000$) has trouble controlling the parameter movement, and it is likely to

get trapped at a stationary point with a singular Jacobian matrix, leading to a large loss despite the convergence. However, this is not an issue in our training regime.



(a) Our training regime

(b) Regular training

(c) $\lambda_{\min}(J(w^*; v^*))$in our training regime

(d) $\lambda_{\min}(J(w^*; v^*))$in regular training regime

Figure 5: Synthetic data: training losses and $\lambda_{\min}(J(w^*; v^*))$ in different width & hidden-weight constraint size ϵ.

4.2 Results on R-ImageNet

We further evaluate our theory on (Restricted) ImageNet ([32]), which is a subset of ImageNet with resolution 224×224, and it is widely used in various papers (e.g. [35], [36]). Detailed description can be found in the supplemental material. In both training regimes, experiments are conducted on ResNet-18, which contains 4 CNN blocks and 1 fully connected output layer. In our training regime, we apply all the constraints in problem (5) to the 4-th CNN block & the output layer. As shown in Figure 6, our training regime can reduce the number of parameters in the 4-th block of ResNet-18 [37] by up to 94% while maintaining competitive test accuracy, especially when ϵ is small. In comparison, regular SGD does not perform well in narrow cases. The presented results are averaged over 2 seeds.

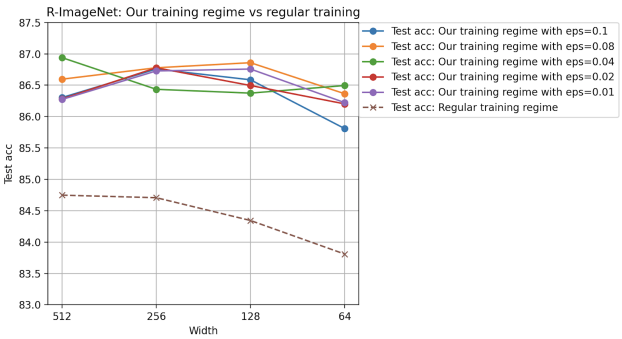


Figure 6: R-ImageNet: Test accuracies under our training regime vs regular training regime.

5 Conclusion

In this work, we shed new light on both the expressivity and trainability of narrow networks. Despite the few neurons, we prove the existence of a strong expressive solution, and the network can be provably trained to approximately zero loss in our training regime. However, our analysis is still limited to 1-hidden-layer networks, and the extension to deep ones will be our future direction.

References

[1] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.

[2] Dawei Li, Tian Ding, and Ruoyu Sun. On the benefit of width for neural networks: Disappearance of bad basins. *arXiv preprint arXiv:1812.11039*, 2018.

[3] Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pages 774–782. PMLR, 2016.

[4] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

[5] Peter Auer, Mark Herbster, Manfred K Warmuth, et al. Exponentially many local minima for single neurons. *Advances in neural information processing systems*, pages 316–322, 1996.

[6] Grzegorz Swirszcz, Wojciech Marian Czarnecki, and Razvan Pascanu. Local minima in training of deep networks. 2016.

[7] Yi Zhou and Yingbin Liang. Critical points of neural networks: Analytical forms and landscape properties. *arXiv preprint arXiv:1710.11205*, 2017.

[8] Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *International Conference on Machine Learning*, pages 2979–2987. PMLR, 2017.

[9] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small nonlinearities in activation functions create bad local minima in neural networks. *arXiv preprint arXiv:1802.03487*, 2018.

[10] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[12] Bo Xie, Yingyu Liang, and Le Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pages 1216–1224. PMLR, 2017.

[13] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[14] 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.

[15] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. arxiv e-prints, art. *arXiv preprint arXiv:1811.08888*, 2018.

[16] 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, 2019.

[17] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *arXiv preprint arXiv:1906.04688*, 2019.

[18] Asaf Noy, Yi Xu, Yonathan Aflalo, Lihi Zelnik-Manor, and Rong Jin. A convergence theory towards practical over-parameterized deep neural networks. *arXiv preprint arXiv:2101.04243*, 2021.

[19] Quynh Nguyen. On the proof of global convergence of gradient descent for deep relu networks with linear widths. *arXiv preprint arXiv:2101.09612*, 2021.

[20] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

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

[22] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

[23] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[24] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[25] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

[26] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *arXiv preprint arXiv:1402.1869*, 2014.

[27] Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

[28] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[29] David Rolnick and Max Tegmark. The power of deeper networks for expressing natural functions. *arXiv preprint arXiv:1705.05502*, 2017.

[30] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pages 3067–3075. PMLR, 2017.

[31] Mo Zhou, Rong Ge, and Chi Jin. A local convergence theory for mildly over-parameterized two-layer neural network. *arXiv preprint arXiv:2102.02410*, 2021.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[33] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

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

[35] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[36] Nathan Inkawhich, Kevin J Liang, Binghui Wang, Matthew Inkawhich, Lawrence Carin, and Yiran Chen. Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability. *arXiv preprint arXiv:2004.14861*, 2020.

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

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] Yes, they do.
   (b) Did you describe the limitations of your work? [Yes] Yes, the limitation is discussed in Section 3.4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Yes, we discuss this part in the first section of the supplemental material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Yes, We have read the ethics review guidelines and we ensure that our paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Yes, we do.
   (b) Did you include complete proofs of all theoretical results? [Yes] Yes, we do, the complete proofs can be seen in the supplemental materials.

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)? [Yes] Yes, all relevant materials are in our supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes, all details are explained in our supplemental materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Yes, in the R-ImageNet experiments, we run experiments on 2 seeds, so the 95 % error bars are vacuous. A meaningful error bar requires multiple repeated experiments, which comes at a cost of heavy computational burden and a very long time span. But still, we provide 60 % error bars in the supplementary material.
   (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)? [Yes] Yes, we introduce the used type of GPU in our supplemental materials.
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? [Yes] Yes, we cited all the creators of the used assets.
   (b) Did you mention the license of the assets? [Yes] Yes, we mentioned all license of datasets in the supplemental material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Yes, we provide our code in the supplemental materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] No, all the data are well-known public benchmark datasets such as MNIST,CIFAR-10, CIFAR-100, R-ImageNet.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] No, all the data are well-known public benchmark datasets such as MNIST,CIFAR-10, CIFAR-100, R-ImageNet.
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]