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, such a solution has a full-rank Jacobian, so it is surrounded by a 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 the 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 ([13, 23, 34]), while narrow networks (width $m <$ sample size n) suffer from highly non-convex landscape with numerous bad local minima ([2, 35, 38, 42, 44]). Another evidence is that pruned models can achieve similar performance to the original model (e.g. [14]), 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 et al. [8].

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:

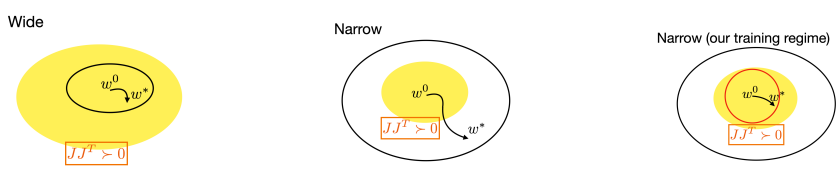
$$\min_{\theta} \ell(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(y_i - f\left(x_i; \theta\right)\right)^2, \tag{2}$$

The gradient of the above problem w.r.t. hidden weights $w = \{w_i\}_{i=1}^m$ 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:

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

When first order methods like (stochastic) gradient descent (GD & SGD) are used, a solution $\theta^* = (w^*, v^*)$ with nearly 0 gradient is returned [5]. 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 ([41], [37]), (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 [11]. 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 ([1, 9, 19, 22, 29, 30, 45, 46]). 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 regimes. 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 [31]. 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. [41], [37]).
- 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 [5].

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, such a solution has a full-rank Jacobian, so 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 (see Figure 1 (c)). 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[2].
- In real-data experiments, 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 [37]. 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. [41] 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. [4]). Particularly, many such works show that certain functions can be efficiently expressible by neural networks with a small number of parameters ([6, 24, 28, 32, 39]), but such a network has to be deep enough. Additionally, their discussions do not involve trainability. Shalev et al. [36] 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. [43] 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.

There are two interesting related works [7, 10] pointed out by the reviewers. Bubeck et al. [7] study how many neurons are required for memorizing a finite dataset by 1-hidden-layer networks. They prove the following results. Their first result is an "existence" result: there exists a network with

[2] For generic constrained optimization, KKT points are expected to be reached by first-order algorithms, but the convergence rates are problem-dependent. In our setting, the iteration complexity of reaching a KKT point will be considered as our future work.

width $m \geq \frac{4n}{d}$ which can memorize n input-label pairs (their Proposition 4). However, in this setting they did not provide an algorithm to find the zero-loss solution. Their second result is related to algorithms: they proposed a training algorithm that achieves accuracy up to error ϵ for a neural net with width $m \geq O\left(\frac{n}{d}\frac{\log(1/\epsilon)}{\epsilon}\right)$. This result requires width dependent on the precision ϵ; for instance, when the desired accuracy $\epsilon = 1/n$, the required width is at least $O\left(\frac{n^2}{d}\right)$. In contrast, in our work, the required number of neurons is just $2n/d$, which is independent of ϵ.

Daniely [10] also studies the expressivity and trainability of 1-hidden-layer networks. To memorize $n(1-\epsilon)$ random data points via SGD, their required width is $O\left((n/d)\left(\log(d\log n)\right)^{\log n}\right)$. With some simple calculations, we find their width is actually at least $O\left(n^2/d\right)$ or even larger for fixed d (see detailed computation & explanation in Appendix A). In contrast, our required width is $2n/d$ for arbitrary n and d. Our bound is always smaller than n when $d > 2$. Additionally, there is a major difference between Daniely [10] and our work: they analyze SGD and we analyze a constrained problem and projected SGD. This may be the reason why we can get a stronger bound on width. In the experiments in Section 4, we observe that SGD performs badly when the width is small (see the first left column in (b), Figure 5). Therefore, we suspect an algorithmic change is needed to train narrow nets with such width (due to the training difficulty), and we indeed propose a new method to train narrow nets.

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

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

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. The idea is that through this initialization, the hidden outputs will cancel out with the outer weights, so that we get zero initial output for any input x; see Fig. 2 for a simple illustration. Note that similar symmetric initialization strategies are also proposed in some recent works such as [9] and [10]. However, our purpose is different. More explanation can be seen in the final paragraph of Section 3.2.

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 $(v_{\frac{m}{2}+1}^0, \ldots, v_m^0) \leftarrow (-v_1^0, \ldots, -v_{\frac{m}{2}}^0)$

Throughout the paper, we will make the following assumptions.

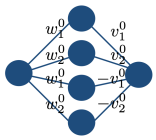


Figure 2: A simple example of the mirrored LeCun's initialization.

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 of (2) with zero loss if it satisfies: $\nabla_w \ell(w^*; v^*) = 0$, $w^* \in B_\epsilon(w^0)$ and v^* is entry-wise non-zero.

Remark 1. *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 sections.*

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.

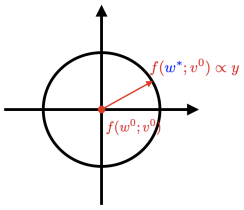
To argue (i), the key idea is to make use of the Inverse (Implicit) Function Theorem (IFT), which is stated in the supplemental material. According to IFT, as long as a $n \times n$ submatrix of $J(w^0; v^0)$ is invertible, then for any $y \in \mathbb{R}^n$, there exists a $w^* \in B_\epsilon(w^0)$ whose 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.

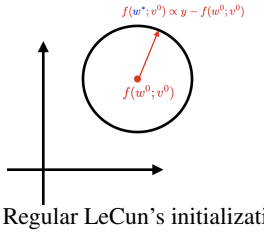
□

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 zero initial output is also used in other recent works in Table 1. Despite the similar design, they use such an initialization for different purposes. In NTK regime, zero initial output helps eliminate the bias term and simplify the proof ([3, 9, 16, 17, 22] and [10]). Nguyen [29] also uses zero 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 zero initial output to identify the strong expressivity of a narrow neural network. To our best knowledge, this is the first time that such zero-output property has been linked to expressivity.

Table 1: Comparison of recent works considering zero initial output.

Work	Width	Motivation
[3, 9, 16, 17, 22]	$m_{L-1} \to \infty$	To avoid handling the bias term in the NTK regime
[10]	$m = \tilde{O}(n/d)$	To avoid handling the bias term in the NTK regime
[29]	$m_{L-1} = O(n)$	To ensure linear convergence via imbalanced weight
Ours	$m = O\left(\frac{n}{d}\right)$	To achieve strong expressivity via 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 [5]. 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 $v=(v_1,\dots,v_{\frac{m}{2}})$ 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 $v=(v_1,\dots,v_{\frac{m}{2}})$, 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

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

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 $v_j/v_{j'}\le\kappa<\infty$ 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:

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

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.

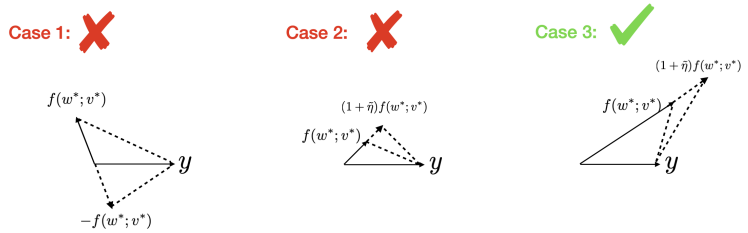


Figure 4: Geometrical illustration of three cases in Lemma 1: comparing with f^*, $f' = (1+\tilde{\eta})f^*$ will not further reduce the distance to y.

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 changing 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 $m \geq \frac{2n}{d}$. Here, we discuss how to extend the results to deep networks. In the proof of Theorem 1, the expressivity is guaranteed if the inputs $\{x_1, \cdots, x_n\}$ follow a *continuous joint distribution*, which is true under Assumption 3. However, the similar result requires a proof for deep networks. Fortunately, we prove this result under a similar condition on the widths, so the expressivity can be shown. More formal analysis is stated and proved 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 as 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.

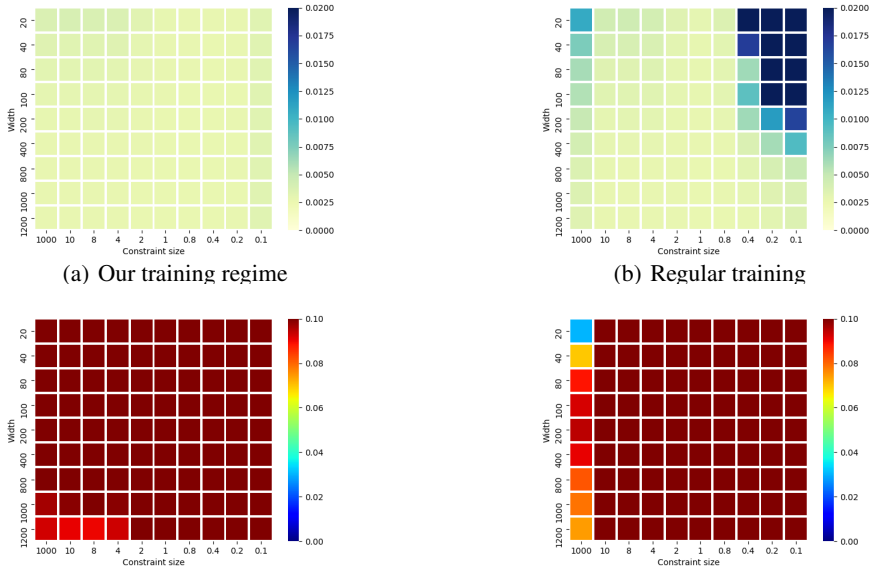
(2) Regular training regime: unconstrained GD-based method with LeCun's initialization.

To reach the KKT point of the constrained problem (5), Projected Gradient Descent (PGD) is used (see Bertsekas [5]), and detailed experimental setup are explained in the supplemental material. Here, we only demonstrate the results on the synthetic data and (Restricted) ImageNet [33]. Experiments on MNIST [21], CIFAR-10 and CIFAR-100 [20] 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

To show the advantage of the proposed method, we conduct experiments on (Restricted) ImageNet ([33]), which is a subset of ImageNet with resolution 224×224, and it is widely used in various papers (e.g. [40], [18]). Detailed description can be found in the supplemental material. In both training regimes, experiments are conducted on ResNet-18 [15], 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 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. These 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] 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.

[2] 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.

[3] Yu Bai and Jason D Lee. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *arXiv preprint arXiv:1910.01619*, 2019.

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

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

[6] 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.

[7] Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, and Dan Mikulincer. Network size and weights size for memorization with two-layers neural networks. *arXiv preprint arXiv:2006.02855*, 2020.

[8] 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.

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

[10] Amit Daniely. Neural networks learning and memorization with (almost) no over-parameterization. *arXiv preprint arXiv:1911.09873*, 2019.

[11] 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.

[12] Kai-Seng Chou et. al. Chapter 4: Inverse function theorem. `https://www.math.cuhk.edu.hk/course_builder/1415/math3060/Chapter%204.%20Inverse%20Function%20Theorem.pdf`. 2015.

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

[14] 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.

[15] 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.

[16] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. *arXiv preprint arXiv:1905.11368*, 2019.

[17] Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *arXiv preprint arXiv:2006.14599*, 2020.

[18] 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.

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

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

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

[22] 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.

[23] 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.

[24] 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.

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[26] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[27] Boris Mityagin. The zero set of a real analytic function. *arXiv preprint arXiv:1512.07276*, 2015.

[28] 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.

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

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

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

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

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

[42] 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.

[43] 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.

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

[45] 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.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [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]

Appendix

Potential Negative Societal Impacts

In this paper, we discuss the expressivity and trainability of narrow neural networks. This paper provides a new understanding from the theoretical study, and such new insights will inspire a better training method for neural networks with a small number of parameters. In industrial applications, several aspects of impact can be expected: training a huge neural network is at the expense of heavy computational burdens and large power consumption, which is a big challenge for embedded systems and small portable devices. In this sense, our work sheds new light on training narrow networks with much fewer parameters, so it will save energy for AI industries and companies. On the other hand, if everyone can afford to train powerful neural networks on their cell phone, then it will be a potential threat to most famous companies & institutes who are boast of their exclusive computational advantages. Additionally, there are chances that neural networks will be used for illegal usage.

A More Discussion on the Related Work: Daniely [10]

Daniely [10] studies the expressivity of 1-hidden-layer networks. They provide two results under different scenarios: to memorize $n(1-\epsilon)$ random input-binary-label pairs via SGD, the required width is either

(i) $\tilde{O}\left(\frac{n}{\epsilon^2}\right)$ ([10, Theorem 5]); or

(ii) $\tilde{O}(n/d)$ ([10, Theorem 7]), where $\tilde{O}$ hides a factor that is dependent on n and d.

In this section, we will explain our claim in Section 2 that their required width can be much larger than ours.

Major differences In our work, our required width is smaller than n (when $d > 2$). However, in either result (i) or (ii), their required width is often much larger than n. In fact, their width is actually at least $O\left(n^2\right)$ or larger for fixed d, if we consider the effect of ϵ (for their first bound) or the hidden factor in $\tilde{O}$ (for their second bound).

We will elaborate below.

For their result (i), similar to Bubeck et al. [7], their required width grows with ϵ. As the authors pointed out before their Theorem 7 on page 8, the required width is $O(n^3)$ if $\epsilon \leq 1/d$. This is why they add result (ii) in [10, Theorem 7].

For their result (ii), i.e., [10, Theorem 7] their required width is roughly $O(n/d\,(\log(d\log n))^{\log n})$. Compared to the desired bound $O(n/d)$, their actual bound contains an additional factor of roughly $(\log(d\log n))^{\log n}$. We would like to stress that the additional factor is *not* a constant factor or a log-factor, but more like a "super-polynomial-factor". As a result, the required width is actually at least $O(n^2/d)$ or larger. The detailed explanations are provided next, and briefly summarized below.

- In Appendix A.1, we will explain why there exists an extra "$(\log(d\log n))^{\log n}$" term in their required width bound. In [10, Theorem 7], their statement only mentioned $\tilde{O}(n/d)$ but not the exact expression.
- In Appendix A.2, we will explain why their bound is at least $O(n^2)$ or larger for fixed d.
- In Appendix A.3, we will summarize some other differences.

A.1 Identifying a More Precise Width Bound

To find a more precise width bound of [10, Theorem 7], we tracked the proof of [10] as follows (note that their width q is our m and their sample size m is our n, and we will use our notation below).

The desired result. Their goal is to memorize the $n(1-\epsilon)$ random input-label pairs via SGD.

Notation. They consider binary labels, i.e., $y_i = \{+1, -1\}$ for $i = 1, \cdots, n$. They consider a feature vector $\Psi_{\boldsymbol{\omega}}(\mathbf{x}_i) \in \mathbb{R}^{dm}$, where $\boldsymbol{\omega} \in \mathbb{R}^{d\times m}$. The predictor (classifier) is in the form of $f_{\boldsymbol{\omega},\mathbf{v}}(\mathbf{x}) := \mathbf{v}^T \Psi_{\boldsymbol{\omega}}(\mathbf{x}_i)$, where $\mathbf{v} \in \mathbb{R}^{dm}$.

Step 1 The desired result holds if there exists $\mathbf{v}$ such that a certain condition holds.

More specifically, they have shown that the desired result holds if the following claim holds: there exists certain $\mathbf{v}$, such that w.h.p. over the dataset and $\boldsymbol{\omega} \in \mathbb{R}^{d\times m}$,

$$\langle \mathbf{v}, \Psi_{\boldsymbol{\omega}}(\mathbf{x}_i)\rangle = y_i + o(1), \quad \forall i = 1, \cdots, n. \tag{A.1}$$

Thus, the goal becomes to find $\mathbf{v}$ such that (A.1) holds. This goal is stated in the first paragraph of Section 4.3, page 15. Though they did not define it explicitly, $o(1)$ in (A.1) means " a constant smaller than 1".

The relation between (A.1) and the desired result is independent of the calculation of the width bound. Anyhow, for completeness, we briefly explain why (A.1) leads to the desired result (most readers can skip the paragraph). First, (A.1) implies that f memorized the dataset for this $(\mathbf{v}, \Psi_{\boldsymbol{\omega}})$. In fact, if (A.1) holds, then $f_{\boldsymbol{\omega},\mathbf{v}}(\mathbf{x}_i)$ has the same sign as $y_i + o(1)$. Note that $y_i + o(1)$ shares the same sign as y_i because y_i is assumed to be either $+1$ or -1 and $o(1)$ is a constant smaller than 1. Together we conclude that $f_{\boldsymbol{\omega},\mathbf{v}}(\mathbf{x}_i)$ shares the same sign as y_i for any i, which means the predictions $\text{sign}(f_{\boldsymbol{\omega},\mathbf{v}}(\mathbf{x}_i)) = y_i$. Second, such a $\mathbf{v}$ can be reached approximately by SGD (see Algorithm 2 on page 10 of [10]). This requires an argument that we skip here.

Step 2: There exists $\mathbf{v}$ which satisfies a relation specified below.

More specifically, they proved the following result.

Theorem A.1 (Theorem 16, page 16, Daniely [10]). *W.p. $1 - \delta - 2^{-\Omega(d)}$ over the choice of dataset and $\boldsymbol{\omega}$, there exists a $\mathbf{v}$ such that the following for all i:*

$$\langle \mathbf{v}, \Psi_{\boldsymbol{\omega}}(\mathbf{x}_i)\rangle = f_{\boldsymbol{\omega}}(\mathbf{x}_i) = y_i + O\left(\frac{(\log(d/\delta))^{\frac{c'}{2}}}{d}\right) + O\left(\sqrt{\frac{d^{c-1}(\log(d/\delta))^{c'+2}}{m}}\right). \tag{A.2}$$

Note that in Theorem A.1, they did not explicitly write out the expression of c and c'. The definition of these two constants can be seen in Section 3.2 and Section 4.3 of A.1 respectively. We will discuss them in Step 3.

Step 3: Identifying a condition on m so that the obtained bound (A.2) in Step 2 implies the desired bound (A.1) in Step 1.

We demonstrate the detailed derivation next.

Step 3.0: To achieve the goal of (A.1), the 3rd term of (A.2) needs to be no more than 1. This term is the crucial part to derive the width bound on m. We provide detailed analysis as follows.

Step 3.1: Now we need to make sure the 3rd term of (A.2) is no more than 1, which is equivalent to (ignoring constant-factors)

$$m \geq d^{c-1}(\log(d/\delta))^{(c'+2)}, \tag{A.3}$$

where $c' \geq 4c + 2$. This definition of c' can be seen in the first paragraph of Section 4.3 on page 15; c, δ are specified in the next two steps.

Step 3.2: Identify $\delta = 1/\log n$. This is specified in the paragraph below Theorem 16, page 16. Plugging it into (A.3), we have

$$m \geq d^{c-1}(\log(d\log n))^{(c'+2)}. \tag{A.4}$$

Step 3.3: Identify $c = \log n/\log d$. The definition of c first appeared in the first paragraph of Section 3.2, where they assume the number of samples is $n = d^c$. This definition is used in the first paragraph of Section 4.3, with a slightly different form $n/d = d^{c-1}$. This definition implies $c = \log n/\log d$. Further, we have

$$c' \geq 4c + 2 \geq 4c + 2 = 4\log n/\log d + 2 \tag{A.5}$$

Plugging (A.5) and $d^{c-1} = n/d$ into (A.4), we have:

$$m \geq \frac{n}{d}(\log(d\log n))^{4\log n/\log d + 2}. \tag{A.6}$$

Finally, ignoring the numerical constants, the bound becomes

$$m \geq O(\frac{n}{d}(\log(d \log n))^{\log n / \log d}). \tag{A.7}$$

A.2 Why The Bound is "Super-Polynomial"

Now we explain why their bound (A.7) is at least $O(n^2)$ or larger for fixed d. The bound (A.7) is a bit complicated as it depends on both d and n. We are more interested in its dependence on n, thus we fix d and analyze how it scales with n. With fixed d, the exponent $\log n / \log d$ can be simplified to $\log n$, thus the bound (A.7) can be simplified to

$$m \geq O(\frac{n}{d}(\log(d \log n))^{\log n}). \tag{A.8}$$

We will show that this bound (A.7) is at least $O(n^2)$ or larger for fixed d.

Define $B_1 = (\log d)^{\log n}$ and $B_2 = [\log(\log n)]^{\log n}$. From (A.8) we obtain

$$m \geq O(\frac{n}{d} \max\{B_1, B_2\}). \tag{A.9}$$

The extra factor $\max\{B_1, B_2\}$ is *not* a constant factor or a log-factor, but more like a "super-polynomial" factor on n, as explained below.

- For B_1, consider the fixed input dimension d for two cases.
  - For d satisfying $\log d > 2.7$ (i.e. $d > 15$), their required width is actually $O\left(\frac{n^2}{d}\right)$. This is an order of magnitude larger than our bound $O(n/d)$.
  - For d satisfying $\log d > 2.7^2$ (i.e. $d > 1395$), the required width is actually $O\left(\frac{n^3}{d}\right)$, two orders of magnitude larger than $O(n/d)$.
- For B_2, when $n > 2.7^{2.7^{27}} \approx 2.2 \times 10^6$, we have $\left(\log(\log(n))^{\log n} > n\right.$. As a result, the required width is at least $O\left(\frac{n^2}{d}\right)$.

Theoretically speaking, it is not hard to prove that their required width can be larger than n^k for any fixed integer k (which is why we say their bound is "super-polynomial"). Empirically speaking, a calculation of their bound for a real dataset can reveal how large it is. On CIFAR-10 dataset [20], $n = 50000, d = 3072$. Plugging these numbers into (A.7) (ignore $O(\cdot)$) we obtain $m \geq 58290499136 >> n$. In comparison, our required width is only $m \geq 2n/d \approx 33$, which is much smaller than $n = 50000$.

In summary, rigorously speaking, their required width $\tilde{O}(n/d)$ is not $O(n/d)$, but can be larger than $O\left(\frac{n^2}{d}\right)$ or even $O\left(\frac{n^3}{d}\right)$.

A.3 Other Differences

Besides the above discussion, there are some other differences between Daniely [10] and our work.

First, they analyze SGD, and we analyze a constrained optimization problem and projected SGD. This may be the reason why we can get a stronger bound on width. In the experiments in Section 4, we observe that SGD performs badly when the width is small (see the first left column in (b), Figure 5). Therefore, we suspect an algorithmic change is needed to train narrow nets with such width (due to the training difficulty), and we indeed propose a new method to train narrow nets.

Second, they consider binary $\{+1, -1\}$ dataset, while our results apply to arbitrary labels. In addition, their proof seems to be highly dependent on the fact that the labels are $\{+1, -1\}$, and seems hard to generalize to general labels.

B Definition and Notations

Before going through the proof details, we restate some of the important notations that will repeatedly appear in the proof, the following notations are also introduced in Section 3.1.

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. To learn such a neural network, we search for the optimal parameter $\theta = (w, v)$ by minimizing the empirical loss:

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

Sometimes we also use $\ell(w;v)$ or $f(w;x,v)$ to emphasize the role of w.

We use the following shorthanded notations:

- $x := (x_1^T; \ldots; x_n^T) \in \mathbb{R}^{n\times d}$, $y := (y_1, \ldots, y_n)^T \in \mathbb{R}^n$;
- $w := (w_1, \ldots, w_m)_{j=1}^m \in \mathbb{R}^{d\times m}$, $v := (v_1, \ldots, v_m)_{j=1}^m \in \mathbb{R}^m$;
- $w_{a,b}$ indicates the b-th component of $w_a \in \mathbb{R}^d$;
- $\theta^0 = (w^0, v^0)$ indicates the initial parameters. Unless otherwise stated, it means the parameters at *the mirrored* LeCun's initialization given in Algorithm 1;
- $f(w;v) := (f(x_1;w,v), f(x_2;w,v), \ldots, f(x_n;w,v))^T \in \mathbb{R}^n$, indicating the neural network output on the whole dataset x.
- Define $\ell \circ f = \frac{1}{2}\|y - f\|_2^2$.

We denote the Jacobian matrix of $f(w;v)$ w.r.t w as

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

We define the feature matrix

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

C Proof of Theorem 1

Theorem 1 consists of two arguments: when the hidden weights w is in the neighborhood of the mirrored LeCun's initialization, we have (i) There exists a global-min with 0 loss, (ii) every stationary point is a global-min. We prove these two arguments respectively. The first part (argument (i)) can be seen in Appendix C.1, the second part (argument (ii)) can be seen in Appendix C.2.

C.1 Proof of the first part of Theorem 1

To prove the first part of Theorem 1, we use the Inverse Function Theorem (IFT) [12] at the mirrored LeCun's initialization. IFT is stated below.

Theorem C.1 (Inverse function theorem (IFT))**.** *Let $\psi : U \to \mathbb{R}^n$ be a C^1 -map where U is open in $\mathbb{R}^n$ and $w \in U$. Suppose that the Jacobian $J\left(w\right)$ is invertible. There exist open sets W and F containing w and $\psi\left(w\right)$ respectively, such that the restriction of ψ on W is a bijection onto F with a C^1 -inverse.*

Our overall proof idea is as follows: we will use IFT to show that: 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. More details can be seen as follows.

In our case, let $\psi = f(w;v^0)$ be the function of w, mapping from $\mathbb{R}^{md}$ to $\mathbb{R}^n$. It may appears that IFT cannot be directly applied since $md \geq 2n$ (cf. Assumption 1), so $f(w;v^0)$ is not dimension-preserved mapping. However, this issue can be alleviated by applying the IFT to *a subvector* of w, while fixing the rest of the variables.

More specifically, we denote $n = k_1 d + k_2$ with $k_1, k_2 \in \mathbb{N}$, and $w = (\tilde{w}^T, \tilde{w}'^T)^T$, where $\tilde{w} = (w_1^T, \cdots, w_{k_1}^T, w_{k_1+1,1}, \cdots, w_{k_1+1,k_2})^T \in \mathbb{R}^n$ and $\tilde{w}' = (w_{k_1+1,k_2+1}, \cdots, w_{k_1+1,d}, w_{k_1+2}^T, \cdots, w_m^T)^T \in \mathbb{R}^{md-n}$. Here, $w_{a,b}$ indicates the b-th component of $w_a \in \mathbb{R}^d$.

We now apply IFT to $f(\tilde{w}; v^0, \tilde{w}'^0) \in \mathbb{R}^n$ (this notation views $\tilde{w}$ as the variable and $v^0, \tilde{w}'^0$ are treated as parameters). Firstly, in Lemma C.1, we prove that w.p.1, the corresponding Jacobian matrix $J(\tilde{w}^0; v^0, \tilde{w}'^0) \in \mathbb{R}^{n\times n}$ is of full rank at the mirrored LeCun's initialization, so that the condition for IFT holds. Then, by IFT, there exist open sets W and F containing $\tilde{w}^0$ and $f(\tilde{w}; v^0, \tilde{w}'^0)$ respectively, such that the restriction of $f(\tilde{w}; v^0, \tilde{w}'^0)$ on W is a bijection onto F. Here, we denote ϵ and δ as the radius of W and F, respectively.

Now, since $f(\tilde{w}; v^0, \tilde{w}'^0) = f(w^0; v^0) = 0 \in \mathbb{R}^n$, set F contains all possible directions pointed from the origin. That is to say, for any label vector $y \in \mathbb{R}^n$, we can always scale it using δ, such that $\delta \frac{y}{\|y\|} \in F$, and then, by IFT , there exists a $\tilde{w}^* \in B_\epsilon(\tilde{w}^0) \subset W$ satisfying

$$f(\tilde{w}^*; v^0, \tilde{w}'^0) = \delta \frac{y}{\|y\|}. \tag{C.1}$$

Since $\tilde{w}$ is just the truncated version of w, (C.1) implies: there exists a $w^* \in B_\epsilon(w^0)$, s.t.

$$f(w^*; v^0) = \delta \frac{y}{\|y\|}. \tag{C.2}$$

Now, we scale the outer weight to $v^* = \frac{\|y\|}{\delta} v^0$ and the output will be exactly y, i.e.:

$$f(w^*; v^*) = y.$$

The proof is concluded.

Lemma C.1. *Under Assumption 1, 2 and 3, as a function of w, v and x, $J(\tilde{w}^0; v^0, \tilde{w}'^0) \in \mathbb{R}^{n\times n}$ is of full rank at the mirrored LeCun's initialization, w.p.1..*

Proof of Lemma C.1. Recall the Jacobian matrix of $f(\tilde{w}; v^0, \tilde{w}'^0)$ w.r.t. $\tilde{w}$:

$$J(\tilde{w}^0; v^0, \tilde{w}'^0) := \begin{bmatrix} \nabla_{\tilde{w}} f\left(\tilde{w}^0; x_1, v^0, \tilde{w}'^0\right)^T \\ \vdots \\ \nabla_{\tilde{w}} f\left(\tilde{w}^0; x_n, v^0, \tilde{w}'^0\right)^T \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

We first consider a general case where $k_2 \neq 0$. Since $f(\tilde{w}; x, v, \tilde{w}') = f(w; x, v) = \sum_{j=1}^m v_j \sigma\left(w_j^T x\right)$, taking derivative w.r.t. $\tilde{w}$ yields $J(\tilde{w}^0; v^0, \tilde{w}'^0)$ equals to:

$$\begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_1\right) x_1^T & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_1\right) x_1^T & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_1\right) x_{1,1} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_1\right) x_{1,k_2} \\ & & \vdots & & & \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) x_n^T & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_n\right) x_n^T & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_n\right) x_{n,1} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_n\right) x_{n,k_2} \end{pmatrix}. \tag{C.3}$$

To prove the full-rankness of $J(\tilde{w}^0; v^0, \tilde{w}'^0)$, we need to show that w.p.1, $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) \neq 0$. Here, $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0))$ is an analytic function since the activation function $\sigma(\cdot)$ is analytic (see Assumption 2). Therefore, we borrow an important result of [27, Proposition 0] which states that the zero set of an analytic function is either the whole domain or zero-measure. The result is formally stated as the following lemma under our notation.

Lemma C.2. *Suppose that: as a function of $\tilde{w}$, $\tilde{w}'$, v and x, $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) : \mathbb{R}^{md+m+nd} \to \mathbb{R}$ is a real analytic function on $\mathbb{R}^{md+m+nd}$. If $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0))$ is not identically zero, then its zero set $\Omega = \left\{\tilde{w}^0, v^0, \tilde{w}'^0, x \mid \det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) = 0\right\}$ has zero measure.*

Based on Lemma C.2, in order to prove $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) \neq 0$ w.p.1, we only need to prove it is not identically zero. To do so, we first transform $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0))$ into its equivalent form:

$$(C.3) \quad = \quad \det\left([B_1, \cdots, B_{k_2}, C_{k_2+1}, \cdots, C_d]\right), \tag{C.4}$$

where

$$B_j = \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_1\right) x_{1,j} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_1\right) x_{1,j} \\ & \vdots & \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) x_{n,j} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_n\right) x_{n,j} \end{pmatrix} \in \mathbb{R}^{n \times (k_1+1)}, \quad j = 1, \cdots, k_2,$$

$$C_j = \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_1\right) x_{1,j} & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_1\right) x_{1,j} \\ & \vdots & \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) x_{n,j} & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_n\right) x_{n,j} \end{pmatrix} \in \mathbb{R}^{n \times k_1}, \quad j = k_2+1, \cdots, d.$$

In addition, B_j can be further rewritten as:

$$B_j = \begin{pmatrix} B_{1,j} \\ \vdots \\ B_{k_2,j} \\ D_j \end{pmatrix} \in \mathbb{R}^{n \times (k_1+1)},$$

where for $i = 1, \cdots k_2$:

$$\begin{aligned} B_{i,j} \quad &= \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)(i-1)+1}\right) x_{(k_1+1)(i-1)+1,j} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_{(k_1+1)(i-1)+1}\right) x_{(k_1+1)(i-1)+1,j} \\ & \vdots & \\ v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)i}\right) x_{(k_1+1)i,j} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_{(k_1+1)i}\right) x_{(k_1+1)i,j} \end{pmatrix} \\ &\in \quad \mathbb{R}^{(k_1+1) \times (k_1+1)}, \end{aligned}$$

and

$$\begin{aligned} D_j \quad &= \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)k_2+1}\right) x_{(k_1+1)k_2+1,j} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_{(k_1+1)k_2+1}\right) x_{(k_1+1)k_2+1,j} \\ & \vdots & \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) x_{n,j} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_n\right) x_{n,j} \end{pmatrix} \\ &\in \quad \mathbb{R}^{(n-(k_1+1)k_2) \times (k_1+1)}. \end{aligned}$$

Similarly, C_j can be further rewritten as:

$$C_j = \begin{pmatrix} C_{1,j} \\ \vdots \\ C_{k_2,j} \\ E_j \end{pmatrix} \in \mathbb{R}^{n \times (k_1)},$$

where for $i = 1, \cdots k_2$:

$$\begin{aligned} C_{i,j} \quad &= \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)(i-1)+1}\right) x_{(k_1+1)(i-1)+1,j} & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_{(k_1+1)(i-1)+1}\right) x_{(k_1+1)(i-1)+1,j} \\ & \vdots & \\ v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)i}\right) x_{(k_1+1)i,j} & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_{(k_1+1)i}\right) x_{(k_1+1)i,j} \end{pmatrix} \\ &\in \quad \mathbb{R}^{(k_1+1) \times k_1} \end{aligned}$$

and

$$\begin{aligned} E_j \quad &= \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)k_2+1}\right) x_{(k_1+1)k_2+1,j} & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_{(k_1+1)k_2+1}\right) x_{(k_1+1)k_2+1,j} \\ & \vdots & \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) x_{n,j} & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_n\right) x_{n,j} \end{pmatrix} \\ &\in \quad \mathbb{R}^{(n-(k_1+1)k_2) \times k_1}. \end{aligned}$$

Therefore, we can rewrite $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0))$ as the following form:

$$\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) = \begin{pmatrix} B_{1,1} & \cdots & B_{1,k_2} & C_{1,k_2+1} & \cdots & C_{1,d} \\ & & \vdots & & & \\ B_{k_2,1} & \cdots & B_{k_2,k_2} & C_{k_2,k_2+1} & \cdots & C_{k_2,d} \\ D_1 & \cdots & D_{k_2} & E_{k_2+1} & \cdots & E_d \end{pmatrix}. \tag{C.5}$$

Based on Lemma C.2, in order to prove $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) \neq 0$ w.p.1., we only need to prove that, as a function of $\tilde{w}$, $\tilde{w}'$, v and x, $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0))$ is not identically zero. To do so, we just need to construct a dataset x such that $(C.5) \neq 0$. We construct such $x := (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ in the following way:

(1) For $i = 1, \cdots, (k_1+1)$: $x_{i,j} = \begin{cases} \delta_i, & j = 1 \\ 0, & \text{otherwise} \end{cases}$, where $\delta_i \neq \delta_{i'} \neq 0, \forall i, i'$.

(2) For $i = (k_1+1)+1, \cdots, 2(k_1+1)$: $x_{i,j} = \begin{cases} \delta_i, & j = 2 \\ 0, & \text{otherwise} \end{cases}$, where $\delta_i \neq \delta_{i'} \neq 0$, $\forall i, i'$.

(3) $\cdots$

(4) For $i = (k_2-1)(k_1+1)+1, \cdots, k_2(k_1+1)$: $x_{i,j} = \begin{cases} \delta_i, & j = k_2 \\ 0, & \text{otherwise} \end{cases}$, where $\delta_i \neq \delta_{i'} \neq 0, \forall i, i'$.

(5) For $i = k_2(k_1+1)+1, \cdots, n$: $x_{i,j} = \begin{cases} 1, & j = i - k_1 k_2 \\ 0, & \text{otherwise} \end{cases}$.

Under such a construction, (C.5) becomes the determinant of a block-diagonal matrix:

$$\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) = \det \begin{pmatrix} B_{1,1} & \cdots & 0 & 0 \\ & & \vdots & \\ 0 & \cdots & B_{k_2,k_2} & 0 \\ 0 & \cdots & 0 & E \end{pmatrix}, \tag{C.6}$$

where $E = [E_{k_2+1} \cdots E_d]$ is a square matrix in $\mathbb{R}^{(n-(k_1+1)k_2) \times (n-(k_1+1)k_2)}$. To prove $(C.6) \neq 0$, we need to prove $B_{1,1}, \cdots, B_{k_2,k_2}$ and E are all full rank matrices.

As for the full-rankness of E, thanks to the construction (5), $E = [E_{k_2+1} \cdots E_d]$ now becomes:

$$E_{k_2+1} = \begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)k_2+1}\right) & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_{(k_1+1)k_2+1}\right) \\ 0 & \cdots & 0 \\ & \vdots & \\ 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{(n-(k_1+1)k_2) \times k_1},$$

$$E_{k_2+2} = \begin{pmatrix} 0 & \cdots & 0 \\ v_1^0 \sigma'\left(w_1^{0T} x_{(k_1+1)k_2+2}\right) & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_{(k_1+1)k_2+2}\right) \\ & \vdots & \\ 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{(n-(k_1+1)k_2) \times k_1},$$

and so on so for:

$$E_d = \begin{pmatrix} 0 & \cdots & 0 \\ & \vdots & \\ 0 & \cdots & 0 \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_n\right) \end{pmatrix} \in \mathbb{R}^{(n-(k_1+1)k_2) \times k_1}.$$

Since $md \geq 2n$ and $J(\tilde{w}^0; v^0, \tilde{w}'^0) \in \mathbb{R}^{n\times n}$ is the jacobian w.r.t. the first n components of $w \in \mathbb{R}^{md}$, it only involves $w_1, w_2, \cdots w_{k_1+1}$ and it will not reach beyond $w_{\frac{m}{2}} \in \mathbb{R}^d$. Recall in the mirrored LeCun's initialization, we only copy the 2nd half of w: $(w^0_{\frac{m}{2}+1}, \ldots, w^0_m) \leftarrow (-w^0_1, \ldots, -w^0_{\frac{m}{2}})$, that is to say, $w_1, w_2, \cdots w_{k_1+1}$ are independent Gaussian random variables, unaffected by the copying phase, similarly for $v_1, \cdots, v_{k_1+1}$.

In short, since Gaussian random variables take value 0 on a zero probability measure, and $\sigma(z) = 0$ only happens when $z = 0$ (see Assumption 2), we have $E = [E_{k_2+1} \cdots E_d]$ is full rank w.p.1.

As for the full-rankness of B_{kk}, $k = 1, \cdots, k_2$, we only need to prove B_{11} is invertible, the proof of the rest of B_{kk} are the same.

Under construction (1), we have:

$$B_{1,1} = \begin{pmatrix} v^0_1 \sigma'\left(w^{0T}_1 x_1\right)\delta_1 & \cdots & v^0_{k_1+1}\sigma'\left(w^{0T}_{k_1+1} x_1\right)\delta_1 \\ & \vdots & \\ v^0_1 \sigma'\left(w^{0T}_1 x_{(k_1+1)}\right)\delta_{k_1+1} & \cdots & v^0_{k_1+1}\sigma'\left(w^{0T}_{k_1+1} x_{(k_1+1)}\right)\delta_{k_1+1} \end{pmatrix} \in \mathbb{R}^{(k_1+1)\times(k_1+1)}.$$

Again, since Gaussian random variables take value 0 on a zero probability measure, and $\delta_i \neq 0$ for $i = 1, \cdots, k_1 + 1$, we only need to prove the following $\tilde{B}_{1,1}$ is full rank:

$$\tilde{B}_{1,1} = \begin{pmatrix} \sigma'\left(w^{0T}_1 x_1\right) & \cdots & \sigma'\left(w^{0T}_{k_1+1} x_1\right) \\ & \vdots & \\ \sigma'\left(w^{0T}_1 x_{(k_1+1)}\right) & \cdots & \sigma'\left(w^{0T}_{k_1+1} x_{(k_1+1)}\right) \end{pmatrix} \in \mathbb{R}^{(k_1+1)\times(k_1+1)}.$$

Next, we borrow the following lemma from [23, Proposition 1], which is the restatement under our notation (their original statement applies for deep neural network, here, we restate it for 1-hidden-layer case in Lemma C.3).

Lemma C.3. *Under Assumption 2, given an 1-hidden-layer neural network with width $m \geq n$, Let $\Omega = \{w \mid \operatorname{rank}(\Phi(w)) < \min\{m, n\}\}$, where $\Phi(w)$ is the hidden feature matrix*

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

Suppose there exists a dimension k such that $x_{i,k} \neq x_{i',k}, \forall i \neq i'$, then Ω is a zero-measure set.

To prove the full-rankness of $\tilde{B}_{1,1}$, we regard it as the hidden feature matrix of an 1-hidden-layer neural network equipped with width $m' = k_1 + 1$ and activation function $\sigma'(z)$, which satisfies Assumption 2. In addition, recall $\delta_i \neq \delta_{i'} \neq 0$ for $\forall i, i'$, so $(x_1, \cdots, x_{k_1+1})$ satisfies the condition of Lemma C.3 w.p.1, and the sample size equals to the width $m' = k_1 + 1$. Therefore, all the assumptions are satisfied and Lemma C.3 directly shows that $\tilde{B}_{1,1}$ is invertible w.p.1..

Similarly, with the same proof technique, it can be shown that the rest of $B_{k,k}$ are also invertible w.p.1.. In conclusion, we have constructed a dataset x, such that (C.6) is non-zero w.p.1., which implies $\Omega = \left\{\tilde{w}^0, v^0, \tilde{w}'^0, x \mid \det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) = 0\right\}$ has zero measure by Lemma C.2. In other words, under the joint distribution of $\tilde{w}^0$, v^0, $\tilde{w}'^0$ and x, $J(\tilde{w}^0; v^0, \tilde{w}'^0)$ is invertible w.p.1.. Recall $\tilde{w}^0$, v^0, and $\tilde{w}'^0$ all follow continuous distribution, furthermore, x also follows a continuous distribution (Assumption 3), so $J(\tilde{w}^0; v^0, \tilde{w}'^0)$ is still invertible w.p.1. under the distribution of x, so the whole proof is completed.

When $n = k_1 d$, or equivalently, $k_2 = 0$, things become easier and we just need to change the size of B_{kk} to $\mathbb{R}^{k_1\times k_1}$, and there is no need to consider $C_{i,j}$, D_j and E_j, the rest of the proof is the same, we omit it for brevity. □

C.2 Proof of the second part of Theorem 1

Suppose we have a 1-hidden-layer neural network $f(x;\theta) = \sum_{j=1}^{m} v_j \sigma\left(w_j^T x\right) \in \mathbb{R}$, and let $f(\theta) \in \mathbb{R}^n$ be output of $(x;\theta)$ on the dataset $x = (x_1, \cdots, x_n)$, let $J(w^*; v^*) \in \mathbb{R}^{n \times md}$ be its Jacobian matrix w.r.t. w at the stationary point $\theta^* = (w^*, v^*)$, we have:

$$\nabla_w \ell(\theta^*) = J(w^*; v^*)^T (f(\theta^*) - y) = 0. \tag{C.7}$$

Therefore, as long as we can prove that $J(w^*; v^*)^T \in \mathbb{R}^{md \times n}$ is full column rank, the stationary point θ^* will become a global minimizer with $\ell(\theta^*) = 0$, so the proof is completed.

Now we prove the full-rankness of $J(w^*; v^*)$. Recall in Lemma C.1, we have proved that, as a function of x, $\det(J(\tilde{w}^0; v^0, \tilde{w}'^0)) \neq 0$ w.p.1., where $J(\tilde{w}^0; v^0, \tilde{w}'^0)$ equals to

$$\begin{pmatrix} v_1^0 \sigma'\left(w_1^{0T} x_1\right) x_1^T & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_1\right) x_1^T & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_1\right) x_{1,1} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_1\right) x_{1,k_2} \\ & & \vdots & & & \\ v_1^0 \sigma'\left(w_1^{0T} x_n\right) x_n^T & \cdots & v_{k_1}^0 \sigma'\left(w_{k_1}^{0T} x_n\right) x_n^T & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_n\right) x_{n,1} & \cdots & v_{k_1+1}^0 \sigma'\left(w_{k_1+1}^{0T} x_n\right) x_{n,k_2} \end{pmatrix}, \tag{C.8}$$

Since $\|w^* - w^0\|_F \leq \epsilon$ and the determinant is a continuous function of w, we have $\det(J(\tilde{w}^*; v^0, \tilde{w}'^*)) \neq 0$ (w.p.1.) when ϵ is small. In addition, as we can see from (C.8), $(v_1^0, \cdots, v_{k_1+1}^0)$ are just constant terms in the corresponding columns, so the full-rankness of (C.8) still holds if we change v_j^0 to $v_j^* \neq 0$. In summary, $\det(J(\tilde{w}^*; v^*, \tilde{w}'^*)) \neq 0$ when v^* is entry-wise non-zero, where $J(\tilde{w}^*; v^*, \tilde{w}'^*)$ equals to

$$\begin{pmatrix} v_1^* \sigma'\left(w_1^{*T} x_1\right) x_1^T & \cdots & v_{k_1}^* \sigma'\left(w_{k_1}^{*T} x_1\right) x_1^T & v_{k_1+1}^* \sigma'\left(w_{k_1+1}^{*T} x_1\right) x_{1,1} & \cdots & v_{k_1+1}^* \sigma'\left(w_{k_1+1}^{*T} x_1\right) x_{1,k_2} \\ & & \vdots & & & \\ v_1^* \sigma'\left(w_1^{*T} x_n\right) x_n^T & \cdots & v_{k_1}^* \sigma'\left(w_{k_1}^{*T} x_n\right) x_n^T & v_{k_1+1}^* \sigma'\left(w_{k_1+1}^{*T} x_n\right) x_{n,1} & \cdots & v_{k_1+1}^* \sigma'\left(w_{k_1+1}^{*T} x_n\right) x_{n,k_2} \end{pmatrix}, \tag{C.9}$$

Furthermore, $J(\tilde{w}^*; v^*, \tilde{w}'^*) \in \mathbb{R}^{n \times n}$ is nothing but a $n \times n$ submatrix of $J(w^*; v^*) \in \mathbb{R}^{n \times md}$. Now that $md \geq 2n > n$, $\det(J(\tilde{w}^*; v^*, \tilde{w}'^*)) \neq 0$ implies the full-row-rankness of $J(w^*; v^*)$ (w.p.1.). Thus the whole proof is completed.

D Proof of Theorem 2

The proof of Theorem 2 consists of proving the following claims: under the setting of Theorem 2,

(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. $\|\nabla_w \ell(w; v)\|_2^2 = \Omega\left(l(w^*; v^*)\right)$, so we have $\ell(w^*; v^*) = O(\epsilon^2)$.

Note that the statement of claim (I) is not precise, there are chances that the constant terms will exponentially grow (will be discussed later). Nevertheless, it is just an intermediate result that helps provide a clearer big picture. our final result in claim (II) will be precise.

To prove claim (I), we only need to consider the case when w^* is on the boundary of the constraint $B_\epsilon(w^0)$ (Theorem 2 automatically holds when w^* is in the interior of $B_\epsilon(w^0)$).

Now, suppose w^* is a non-zero-gradient KKT point on the boundary, by the optimality condition, its negative gradient direction should be along the same direction as $w^* - w^0$, therefore, if we further take a small step η along the negative gradient direction, we have:

$$-\eta \nabla_w \ell(w^*; v^*) = \eta \frac{\|\nabla_w \ell(w^*; v^*)\|_2}{\|w^* - w^0\|_2} (w^* - w^0) = \tilde{\eta}(w^* - w^0), \tag{D.1}$$

where $\tilde{\eta} := \eta \frac{\|\nabla_w \ell(w^*;v^*)\|_2}{\|w^*-w^0\|_2} = \eta \frac{\|\nabla_w \ell(w^*;v^*)\|_2}{\epsilon}$. In this case, the loss function will decrease when we further move w along $-\nabla_w \ell(w^*;v^*)$ with a sufficiently small stepsize η. In other words, we can apply Descent lemma (details can be seen in Bertsekas et al. [5]) in this local region, i.e.

$$\ell\left(w^* - \eta \nabla_w \ell(w^*;v^*); v^*\right) - \ell(w^*;v^*) \quad \leq \quad -\eta \|\nabla_w \ell(w^*;v^*)\|_2^2. \tag{D.2}$$

As a matter of fact, after further taking a small GD step at w^*, $f\left(w^* - \eta \nabla_w \ell(w^*;v^*); v^*\right)$ will be closer to the groundtruth y, we will come back to this fact later, it will be used to bound $\|\nabla_w \ell(w^*;v^*)\|_2$.

Now, for any $w \in B_\epsilon(w^0)$, we take the Taylor expansion of $f(w;v^*)$ at w^0:

$$\begin{aligned}
f(w;v^*) &= f(w^0;v^*) + J(w^0;v^*)(w-w^0) + R(w) && \text{(D.3)}\\
&\overset{\text{(a) \& (b)}}{=} J(w^0;v^*)w + R(w), && \text{(D.4)}\\
f_{\text{lin}}(w;v^*) &:= f(w^0;v^*) + J(w^0;v^*)(w-w^0) && \text{(D.5)}\\
&\overset{\text{(a) \& (b)}}{=} J(w^0;v^*)w, && \text{(D.6)}
\end{aligned}$$

where $R(w) \in \mathbb{R}^n$ is the residue term of the Taylor expansion:

$$[R(w)]_i = \int_0^1 (w-w^0)^T H_i(w^0 + t(w-w^0))(w-w^0)(1-t)dt, \quad i = 1, \cdots, n, \tag{D.7}$$

where $H_i(w)$ is the Hessian matrix of $f(w;v^*,x_i)$ at w (for simplicity, we drop the dependence of v^* and x_i in the notation), and $f_{\text{lin}}(w;v^*)$ is a linear approximation of $f(w;v^*)$. In addition, (a) & (b) is due to the fact that w_j^0 follows the mirrored LeCun's initialization with $(w_{\frac{m}{2}+1}^0, \ldots, w_m^0) = (w_1^0, \ldots, w_{\frac{m}{2}}^0)$, recall the construction of $f(w;v)$ in (4), the hidden output will cancel out with the outer weight, so we have (a):

$$f(w^0;v^*) \overset{(4)}{=} \sum_{j=1}^{\frac{m}{2}} v_j^* \left(\sigma(w_j^{0T}x) - \sigma(w_{j+\frac{m}{2}}^{0T}x)\right) = 0. \tag{D.8}$$

Similarly, we have (b):

$$J(w^0;v^*)w^0 = \nabla_w f\left(w^0;x,v^*\right)^T w^0 = \sum_{j=1}^{\frac{m}{2}} v_j^* \left(\sigma'(w_j^{0T}x)w_j^{0T}x - \sigma'(w_{j+\frac{m}{2}}^{0T}x)w_{j+\frac{m}{2}}^{0T}x\right) = 0. \tag{D.9}$$

Based on (D.4), we define the following quantities:

$$\begin{aligned}
\bar{f} &:= f\left(w^* - \eta \nabla_w \ell(w^*;v^*); v^*\right) \overset{(D.4)}{=} J(w^0;v^*)(w^* - \eta \nabla_w \ell(w^*;v^*)) + \bar{R}, && \text{(D.10)}\\
f^* &:= f\left(w^*;v^*\right) \overset{(D.4)}{=} J(w^0;v^*)w^* + R^*, && \text{(D.11)}
\end{aligned}$$

where $\bar{R} = R\left(w^* - \eta \nabla_w \ell(w^*;v^*)\right)$, $R^* = R\left(w^*\right)$ as it is introduced in (D.7). Additionally, $\bar{f}$ can be re-written in the form of f^*:

$$\begin{aligned}
\bar{f} &\overset{(D.10)}{=} J(w^0;v^*)(w^* - \eta \nabla_w \ell(w^*;v^*)) + \bar{R} && \text{(D.12)}\\
&= J(w^0;v^*)w^* - \eta J(w^0;v^*)\nabla_w \ell(w^*;v^*) + \bar{R} && \text{(D.13)}\\
&\overset{(D.11)}{=} f^* - R^* - \eta J(w^0;v^*)\nabla_w \ell(w^*;v^*) + \bar{R} && \text{(D.14)}\\
&\overset{(D.1)}{=} f^* - R^* + \tilde{\eta} J(w^0;v^*)(w^* - w^0) + \bar{R} && \text{(D.15)}\\
&\overset{(D.9)}{=} f^* - R^* + \tilde{\eta} J(w^0;v^*)(w^*) + \bar{R} && \text{(D.16)}\\
&\overset{(D.11)}{=} f^* - R^* + \tilde{\eta}(f^* - R^*) + \bar{R} && \text{(D.17)}\\
&= (1+\tilde{\eta})f^* - (1+\tilde{\eta})R^* + \bar{R}. && \text{(D.18)}
\end{aligned}$$

Now, we construct a rough estimator of $\bar{f}$ by merely adding a residue term $(1+\tilde{\eta})R^*$ on $\bar{f}_{\text{lin}}$, which is f' defined as follows:

$$
\begin{aligned}
f' &:= \bar{f}_{\text{lin}} + (1+\tilde{\eta})R^* && \text{(D.19)}\\
&= f_{\text{lin}}\left(w^* - \eta\nabla_w \ell(w^*;v^*); v^*\right) + (1+\tilde{\eta})R^* && \text{(D.20)}\\
&\overset{\text{(D.6)}}{=} J(w^0;v^*)(w^* - \eta\nabla_w \ell(w^*;v^*)) + (1+\tilde{\eta})R^* && \text{(D.21)}\\
&= J(w^0;v^*)w^* - \eta J(w^0;v^*)\nabla_w \ell(w^*;v^*) + (1+\tilde{\eta})R^* && \text{(D.22)}\\
&\overset{(D.11)}{=} f^* - R^* + \eta J(w^0;v^*)\nabla_w \ell(w^*;v^*) + (1+\tilde{\eta})R^* && \text{(D.23)}\\
&\overset{(D.1)}{=} f^* - R^* + \tilde{\eta} J(w^0;v^*)(w^* - w^0) + (1+\tilde{\eta})R^* && \text{(D.24)}\\
&\overset{(D.9)}{=} f^* - R^* + \tilde{\eta} J(w^0;v^*)(w^*) + (1+\tilde{\eta})R^* && \text{(D.25)}\\
&\overset{(D.11)}{=} f(w^*;v^*) - R^* + \tilde{\eta}(f^* - R^*) + (1+\tilde{\eta})R^* && \text{(D.26)}\\
&= (1+\tilde{\eta})f^*. && \text{(D.27)}
\end{aligned}
$$

According to the descent property in (D.2), after taking a very small GD step at w^* , the new loss function $\ell \circ \bar{f}$ will be smaller than the old one $\ell \circ f^*$, where $\ell \circ f = \frac{1}{2}\|y - f\|_2^2$. In contrast, we discuss the change of the loss function from f^* to that of the rough estimator f'. The following Lemma 1 shows that $\ell \circ f' \geq \ell \circ f^*$, different from the fact that $\ell \circ \bar{f} \leq \ell \circ f^*$.

Lemma D.1. *[Corresponding to Lemma 1 in the full paper.] Under the background of Theorem 2 and the definition of f' & f^* in (D.27) & (D.11), we have $\ell \circ f' \geq \ell \circ f^*$, i.e., $\|f' - y\|_2^2 \geq \|f^* - y\|_2^2$.*

The proof of Lemma D.1 can be seen in Appendix D.1.

Now, we have

$$
\begin{aligned}
\eta \left\|\nabla_w \ell(w^*;v^*)\right\|_2^2 &\overset{(D.2)}{\leq} \ell(w^*;v^*) - \ell\left(w^* - \eta\nabla_w \ell(w^*;v^*); v^*\right) && \text{(D.28)}\\
&= \frac{1}{2}\left\|f^* - y\right\|_2^2 - \frac{1}{2}\left\|\bar{f} - y\right\|_2^2 && \text{(D.29)}\\
&\overset{\text{Lemma1}}{\leq} \frac{1}{2}\left\|f' - y\right\|_2^2 - \frac{1}{2}\left\|\bar{f} - y\right\|_2^2 && \text{(D.30)}\\
&= \frac{1}{2}\left\|f' - \bar{f}\right\|_2 \left\|f' + \bar{f} - 2y\right\|_2. && \text{(D.31)}
\end{aligned}
$$

Next, we bound $\left\|f' - \bar{f}\right\|_2 \left\|f' + \bar{f} - 2y\right\|_2$ using the following Lemma D.2.

Lemma D.2. *[Corresponding to Lemma 2 in the full paper.] Under the background of Theorem 2 and the definition of f' & $\bar{f}$ in (D.27) & (D.10), we have:*

$$
\left\|f' - \bar{f}\right\|_2 \left\|f' + \bar{f} - 2y\right\|_2 \leq \left(\tilde{\eta}\epsilon^2 \lambda_{[0:2]}\sqrt{20n}\right)\left(\sqrt{n}C_y + 3\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}}\right), \tag{D.32}
$$

where ζ is the constraint for v required in $B(v)$: $v \geq \zeta\mathbf{1}$; $\lambda_{[a:b]} := \max_i\{\lambda_{i[a:b]} | i = 1, \cdots, n\}$, and each $\lambda_{i[a:b]}$ is the maximum eigenvalue of the Hessian matrices $H_i(w^0 + t(w^* - w^0)) := \nabla_w^2 f(w^0 + t(w^* - w^0); v^*, x_i)$ in the interval $t \in [a, b]$.

The proof of Lemma D.2 can be seen in Appendix D.2. Now, with the help of Lemma D.2, we have:

$$
\begin{aligned}
\eta \left\|\nabla_w \ell(w^*;v^*)\right\|_2^2 &\overset{(D.31)}{\leq} \frac{1}{2}\left\|f' - \bar{f}\right\|_2 \left\|f' + \bar{f} - 2y\right\|_2 && \text{(D.33)}\\
&\overset{(D.32)}{\leq} \frac{1}{2}\left(\tilde{\eta}\epsilon^2 \lambda_{[0:2]}\sqrt{20n}\right)\left(\sqrt{n}C_y + 3\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}}\right). && \text{(D.34)}
\end{aligned}
$$

Recall $\eta = \frac{\epsilon}{\|\nabla_w \ell(w^*;v^*)\|_2}\tilde{\eta}$ and $\tilde{\eta}$ is sufficiently small, we have:

$$
\left\|\nabla_w \ell(w^*;v^*)\right\|_2 \leq \frac{1}{2}\epsilon\left(\lambda_{[0:2]}\sqrt{20n}\right)\left(\sqrt{n}C_y + 3\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}}\right). \tag{D.35}
$$

Since ζ can be chosen arbitrarily small, we can choose it to be smaller than $\frac{1}{\epsilon}$. That is to say, $\|\nabla_w \ell(w^*; v^*)\|_2 = O(\epsilon)$, so the claim (I) is proved. Note that to be precise, the constant on the right hand side of the above inequality depends on $\lambda_{[0:2]}$, which is a linear function of v, and the latter vector may not be upper bounded. Nevertheless, claim (I) is just an intermediate result, and our subsequent derivation will directly use the right hand side of (D.35), which is precise.

Now, we build the relationship between $\|\nabla_w \ell(w^*; v^*)\|_2$ and the loss function. Here, we need to eliminate the dependence of v in the final result: since there is no uniform upper bound for v, it can potentially make $\lambda_{[0:2]}$ grow exponentially. To alleviate this issue, we utilize the fact that $f(x;\theta)$ is linear in v, so $\lambda_{[0:2]}$ is also linear in v, and then we manage to remove the dependence of v in our final result. Specifically, let us define

$$v^*_{\text{min}} := \min_j \{v^*_j \mid j = 1, \cdots, m\}, \quad v^*_{\text{max}} := \max_j \{v^*_j \mid j = 1, \cdots, m\}.$$

So we have

$$\begin{aligned} \|\nabla_w \ell(w^*; v^*)\|_2^2 &= \|J(w^*; v^*)^T (y - f^*)\|_2^2 \overset{(c)}{\geq} v^{*2}_{\text{min}} \cdot \|\tilde{J}(w^*; v^*)^T (y - f^*)\|_2^2 \quad \text{(D.36)} \\ &\geq v^{*2}_{\text{min}} \cdot \tilde{\lambda}^{*2}_{\text{min}} \cdot \|y - f^*\|_2^2, \quad \text{(D.37)} \end{aligned}$$

where $\tilde{\lambda}^*_{\text{min}}$ is the smallest singular value of $\tilde{J}(w^*; v^*)$, and $\tilde{J}(w^*; v^*)$ is:

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

Here, (c) is straightforward because $\tilde{J}(w^*; v^*)$ is just the simplified version of $J(w^*; v^*)$ by removing all the coefficient v^*_j; furthermore, $\tilde{J}(w^*; v^*)$ is full row rank because (i) $J(w^*; v^*)$ is proved to be full rank in the second part of Theorem 1 in Appendix C.2, (ii) v^* is entry-wise non-zero, so $\tilde{\lambda}^*_{\text{min}}$ is strictly positive.

Now, we need to remove the dependence of v in the right hand side of (D.35). Similarly as before, $\lambda_{[a:b]}$ can also be bounded by $v^*_{max} \tilde{\lambda}_{[a:b]}$, where $\tilde{\lambda}_{[a:b]}$ is equal to $\max_i \{\tilde{\lambda}_{i[a:b]} | i = 1, \cdots, n\}$, and each $\tilde{\lambda}_{i[a:b]}$ is the maximum eigenvalue of the Hessian matrices $\tilde{H}_i(w^0 + t(w^* - w^0)) := \nabla_w^2 \tilde{f}(w^0 + t(w^* - w^0); , x_i)$ in the interval $t \in [a, b]$, and $\tilde{f}(w; , x_i) := \sum_{j=1}^m \sigma(x_i^T w_j)$ is the simplified version of $f(w; v, x_i)$ by removing all the coefficient v_j. In conclusion, we have

$$\begin{aligned} v^*_{\text{min}} \tilde{\lambda}^*_{\text{min}} \|y - f^*\|_2 &\leq \|\nabla_w \ell(w^*; v^*)\|_2 \quad \text{(D.38)} \\ &\leq \frac{1}{2}\epsilon \left(\lambda_{[0:2]} \sqrt{20n}\right) \left(\sqrt{n} C_y + 3\left(n\left(\frac{m}{2} L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}}\right) \quad \text{(D.39)} \\ &\leq \frac{1}{2}\epsilon \left(v^*_{\text{max}} \tilde{\lambda}_{[0:2]} \sqrt{20n}\right) \left(\sqrt{n} C_y + 3\left(n\left(\frac{m}{2} L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}}\right). \quad \text{(D.40)} \end{aligned}$$

Rearrange and take the square on both sides, we get $\|y - f^*\|_2^2 = O\left(\kappa^2 \epsilon^2\right)$, where κ is the finite constant in the constraint $B(v)$ of problem (5). So $\ell(\theta^*) = O(\epsilon^2)$, the proof of Theorem 2 is completed.

D.1 Proof of Lemma 1

To prove Lemma 1, we need to discuss the following cases:

(i) When $f^{*T} y \geq 0$, we prove Lemma 1 by contradiction: when $\tilde{\eta}$ is sufficiently small, suppose $\|(1+\tilde{\eta}) f^* - y\|_2^2 < \|f^* - y\|_2^2$, then (w^*, v^*) is not a KKT point (this case corresponds to the Figure 4 (Middle)). Note that in problem (5), $f^* = \sum_{j=1}^{\frac{m}{2}} v_j^* \left(\sigma(w_j^{*T} x) - \sigma(w_{j+\frac{m}{2}}^{*T} x)\right)$ is linear in $(v_1^*, \cdots, v_{\frac{m}{2}}^*)$, so $(1+\tilde{\eta}) f^* = f(w^*; (1+\tilde{\eta}) v^*)$. That is to say, $\|(1+\tilde{\eta}) f^* - y\|_2^2 < \|f^* - y\|_2^2$ implies $\|f(w^*; (1+\tilde{\eta}) v^*) - y\|_2^2 < \|f^* - y\|_2^2$, which means we can further reduce the loss by changing $v^* \to (1+\tilde{\eta}) v^*$. Since $(1+\tilde{\eta}) v^*$ is still feasible if v^* is feasible, we have a contradiction to the assumption that $(w,^* v^*)$ is a KKT point.

Therefore, in case (i), moving from f^* to $f' = (1+\tilde{\eta})f^*$ will not further reduce the loss (this case corresponds to the Figure 4 (Right)). In other words, we always have $f^{*T}(f^* - y) \geq 0$, this property will also be used in Lemma D.2.

(ii) When $f^{*T}y < 0$, we have

$$\begin{aligned}
\|f^* - y\|_2^2 &= \|\Phi(w^*)v^* - y\|_2^2 && \text{(D.41)}\\
&= \|\Phi(w^*)v^*\|_2^2 + \|y\|_2^2 - 2(\Phi(w^*)v)^T y && \text{(D.42)}\\
&> \|f^*\|_2^2 + \|y\|_2^2 - 2(-\Phi(w^*)v)^T y && \text{(D.43)}\\
&= \|\Phi(w^*)(-v^*) - y\|_2^2 && \text{(D.44)}\\
&= \|-f^* - y\|_2^2, && \text{(D.45)}
\end{aligned}$$

where

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

Therefore, changing v^* to $-v^*$ will further reduce the loss function (see Figure 4 (Left)). Since $-v^*$ is feasible if v^* is feasible, this is a contradiction to the assumption that (w^*, v^*) is a KKT point. That is to say, we always have case (i): $f^{*T}y \geq 0$.

In conclusion, we always have $\ell \circ f' \geq \ell \circ f^*$, so the proof is completed.

D.2 Proof of Lemma 2

Similarly with the residue term (D.7), we define $R \in \mathbb{R}^n$ with each component satisfying $[R]_i := \int_0^{1+\tilde{\eta}} (w - w^0)^T H_i(w^0 + t(w - w^0))(w - w^0)(1-t)dt$, we have:

$$\begin{aligned}
\left\|f' - \bar{f}\right\|_2^2 &\overset{(D.27)\&(D.10)}{=} \left\|-(1+\tilde{\eta})R^* + \bar{R}\right\|_2^2 && \text{(D.46)}\\
&= \left\|(1+\tilde{\eta})R^* - \bar{R}\right\|_2^2 && \text{(D.47)}\\
&\overset{(*)}{=} \left\|(1+\tilde{\eta})R^* - (1+\tilde{\eta})^2 R\right\|_2^2 && \text{(D.48)}\\
&= \|(1+\tilde{\eta})(R^* - R) - (1+\tilde{\eta})\tilde{\eta}R\|_2^2 && \text{(D.49)}\\
&\leq (1+\tilde{\eta})^2 \sum_{i=1}^n \left(\int_1^{1+\tilde{\eta}} (w - w^0)^T H_i(w^0 + t(w - w^0))(w - w^0)(1-t)dt\right)^2 \\
&\quad + (1+\tilde{\eta})^2\tilde{\eta}^2 \sum_{i=1}^n \left(\int_0^{1+\tilde{\eta}} (w - w^0)^T H_i(w^0 + t(w - w^0))(w - w^0)(1-t)dt\right)^2 \\
&\leq (1+\tilde{\eta})^2 \sum_{i=1}^n \left(\lambda_{i[1:1+\tilde{\eta}]}\epsilon^2 \int_1^{1+\tilde{\eta}} dt\right)^2 + (1+\tilde{\eta})^2\tilde{\eta}^2 \sum_{i=1}^n \left(\lambda_{i[0:1+\tilde{\eta}]}\epsilon^2 \int_0^{1+\tilde{\eta}} dt\right)^2 \\
&= n(1+\tilde{\eta})^2\tilde{\eta}^2\lambda_{[1:1+\tilde{\eta}]}^2\epsilon^4 + n(1+\tilde{\eta})^4\tilde{\eta}^2\lambda_{[0:1+\tilde{\eta}]}^2\epsilon^4 && \text{(D.50)}\\
&= n\tilde{\eta}^2\epsilon^4\left((1+\tilde{\eta})^2\lambda_{[1:1+\tilde{\eta}]}^2 + (1+\tilde{\eta})^4\lambda_{[0:1+\tilde{\eta}]}^2\right) && \text{(D.51)}\\
&= n\tilde{\eta}^2\epsilon^4\left(4\lambda_{[1:1+\tilde{\eta}]}^2 + 16\lambda_{[0:1+\tilde{\eta}]}^2\right) && \text{(D.52)}\\
&= n\tilde{\eta}^2\epsilon^4\left(20\lambda_{[0:2]}^2\right), && \text{(D.53)}
\end{aligned}$$

where the last two inequalities is because of the fact that $\tilde{\eta} \leq 1$ is sufficiently small and $\lambda_{[1:1+\tilde{\eta}]} \leq \lambda_{[0:1+\tilde{\eta}]} \leq \lambda_{[0:2]}$. (*) is due to: for $i = 1, \cdots, n$:

$$\begin{aligned}
[\bar{R}]_i &= \int_0^1 \left(w - \eta\nabla_w \ell(w^*; v^*) - w^0\right)^T H_i\left(w^0 + t(w - \eta\nabla_w \ell(w^*; v^*) - w^0)\right)\left(w - \eta\nabla_w \ell(w^*; v^*) - w^0\right)(1-t)dt && \text{(D.54)}\\
&\overset{(D.1)}{=} (1+\tilde{\eta})^2 \int_0^1 \left(w - w^0\right)^T H_i\left(w^0 + (1+\tilde{\eta})t(w - w^0)\right)\left(w - w^0\right)(1-t)dt && \text{(D.55)}\\
&= (1+\tilde{\eta})^2 \int_0^{1+\tilde{\eta}} \left(w - w^0\right)^T H_i\left(w^0 + t(w - w^0)\right)\left(w - w^0\right)(1-t)dt && \text{(D.56)}\\
&= (1+\tilde{\eta})^2 [R]_i. && \text{(D.57)}
\end{aligned}$$

Now, we bound $\left\|f' + \bar{f} - 2y\right\|_2$, since (w^*, v^*) is a KKT point, it is proved in Lemma D.1 in Appendix D.1 that $f^{*T}y \geq 0$, furthermore, at $w = w^*$, the loss function at (w^*, v^*) should be less or equal to all other feasible points (w^*, v), including $v = \zeta\mathbf{1}$, i.e.,

$$\begin{aligned}
\|f^* - y\|_2^2 &= \|f(w^*; v^*) - y\|_2^2 && \text{(D.58)}\\
&\leq \|f(w^*; v^*)\|^2 + \|y\|_2^2 && \text{(D.59)}\\
&\leq \|f(w^*; \zeta\mathbf{1})\|^2 + \|y\|_2^2 && \text{(D.60)}\\
&\overset{(**)}{\leq} n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + \|y\|_2^2 && \text{(D.61)}\\
&\leq n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2, && \text{(D.62)}
\end{aligned}$$

where the last inequality is because of Assumption 3 (each $y_i \leq C_y$), and (**): is due to

$$\begin{aligned}
\|f(w; \zeta\mathbf{1})\|_2^2 &\overset{(4)}{=} \sum_{i=1}^{n}\left(\sum_{j=1}^{\frac{m}{2}}\zeta\left(\sigma(w_j^T x_i) - \sigma(w_{j+\frac{m}{2}}^T x_i)\right)\right)^2 && \text{(D.63)}\\
&= \zeta^2\sum_{i=1}^{n}\left(\sum_{j=1}^{\frac{m}{2}}\left(\sigma(w_j^T x_i) - \sigma(w_{j+\frac{m}{2}}^T x_i)\right)\right)^2 && \text{(D.64)}\\
&\overset{\text{Assumption 2}}{\leq} \zeta^2 L^2\sum_{i=1}^{n}\left(\sum_{j=1}^{\frac{m}{2}}(w_j - w_{j+\frac{m}{2}})^T x_i\right)^2 && \text{(D.65)}\\
&\leq \zeta^2 L^2\sum_{i=1}^{n}\left(\sum_{j=1}^{\frac{m}{2}}\|w_j - w_{j+\frac{m}{2}}\|_2\|x_i\|_2\right)^2 && \text{(D.66)}\\
&\overset{(***)}{=} \zeta^2 L^2\sum_{i=1}^{n}\left(\sum_{j=1}^{\frac{m}{2}}\|w_j - w_{j+\frac{m}{2}}\|_2\right)^2 && \text{(D.67)}\\
&\leq n\left(\frac{m}{2}L\zeta\epsilon\right)^2, && \text{(D.68)}
\end{aligned}$$

where $(***)$: we assume $\|x_i\|_2 \leq 1$ for $i = 1, \cdots, n$. For general $\|x_i\|_2$, the difference is up to a constant.

Recall $f' = (1 + \tilde{\eta})f^*$, we have

$$\begin{aligned}
\|f' - y\|_2 &= \|(1 + \tilde{\eta})f^* - y\|_2 && \text{(D.69)}\\
&= \|f^* - y + \tilde{\eta}(f^* - y) + \tilde{\eta}y\|_2 && \text{(D.70)}\\
&= \|f^* - y\|_2 + \tilde{\eta}\|f^* - y\|_2 + \tilde{\eta}\|y\|_2 && \text{(D.71)}\\
&\overset{(D.62)}{\leq} (1 + \tilde{\eta})\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}} + \tilde{\eta}\|y\|_2 && \text{(D.72)}\\
&\leq (1 + \tilde{\eta})\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}} + \tilde{\eta}\sqrt{n}C_y && \text{(D.73)}\\
&\leq 2\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}} + \sqrt{n}C_y, && \text{(D.74)}
\end{aligned}$$

where the last inequality is because $\tilde{\eta}$ is sufficiently small. Now, combining with the descent property $\|\bar{f} - y\|_2 \leq \|f^* - y\|_2$, we have

$$\begin{aligned}
\|f' + \bar{f} - 2y\|_2 &\leq \|f' - y\|_2 + \|\bar{f} - y\|_2 && \text{(D.75)}\\
&\leq \|f' - y\|_2 + \|f^* - y\|_2 && \text{(D.76)}\\
&\leq 3\left(n\left(\frac{m}{2}L\zeta\epsilon\right)^2 + nC_y^2\right)^{\frac{1}{2}} + \sqrt{n}C_y. && \text{(D.77)}
\end{aligned}$$

We conclude the proof of Lemma 2 by combining (D.53) and (D.77).

E Extension to Deep Networks

In this section, we discuss how to extend our analysis to deep networks . To do so, 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. In the proof of Theorem 1, the expressivity is guaranteed if the inputs $\{x_1, \cdots, x_n\}$ follow a *continuous joint distribution*, which is true under Assumption 3. 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 outputs of the $(L-2)$-th layer still follow a *continuous joint distribution* under Assumption 2 and 3, so the expressivity can be shown using the similar technique as Theorem 1. This result is formally stated and proved in the Lemma E.1 below.

Lemma E.1. *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, $w^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}}$ are the weights, $l = 1, \ldots, L$. Under Assumption 2 and 3, suppose $m_l \geq m_{l+1}$, for $l \leq L-3$ and $m_{L-1}m_{L-2} \geq 2n$, then at the initialization θ^0 (for $l \leq L-2$, LeCun's initialization is used; for the last two layers, the mirrored LeCun's initialization is used), for the inputs $\{x_1, \cdots, x_n\}$, the outputs of the $(L-2)$-th layer follow a continuous joint distribution.

To prove Lemma E.1, we first prove the following lemma:

Lemma E.2. *Suppose that $\psi : \mathbb{R}^{k_1} \to \mathbb{R}^{k_2}$ is a analytic mapping and for almost every $u \in \mathbb{R}^{k_1}$, the Jacobian matrix $J(u)$ of ψ w.r.t. u is of full row rank. If $u \in \mathbb{R}^{k_1}$ follows a continuous distribution and $k_1 \geq k_2$, then $\psi(u)$ also follows a continuous distribution.*

Proof. Let $Z_0 \subseteq \mathbb{R}^{k_2}$ be a zero measure set in $\mathbb{R}^{k_2}$, We define $S_1(Z_0) = \{u \in \mathbb{R}^{k_1} \mid \psi(u) \in Z_0, J(u) \text{ is non-singular}\}$. By the definition of $S_1(Z_0)$, any $u \in S_1(Z_0)$ can be written as $u = (u_1^T, u_2^T)^T$, where $u_1 \in \mathbb{R}^{k_2}$ and $J(u_1; u_2)$ is invertible ($J(u_1; u_2)$ is the $l \times l$ submatrix of $J(u)$, similarly as in Lemma C.1). Then by the Inverse Function Theorem, there exists some ball $\mathcal{B}_{\epsilon(u)}(u) \subseteq \mathbb{R}^{k_1}$ (centered at u with radius $\epsilon(u)$) such that for any $u' = ((u')_1^T, (u')_2^T)^T \in \mathcal{B}_{\epsilon(u)}(u) \cap S_1(Z_0)$, $u'_1 = \tau(u'_2, z')$, where $z' = \psi(u') \in Z_0$ and τ is a smooth mapping in a neighborhood $\tilde{Z}_0$ of $(u_2, \psi(u))$. Then for any u, there exists a rational point $\bar{u} \in \mathbb{Q}^{k_1}$ and a rational number $\bar{\epsilon}(u) \in \mathbb{Q}$ such that $u \in N(u) := \mathcal{B}_{\bar{\epsilon}(u)}(\bar{u}) \subseteq \mathcal{B}_{\epsilon(u)}(u)$. Since the collection of all open balls with a rational center and a rational radius is a countable set, we let $N_1, N_2, \cdots, N_n, \cdots$ be different $N(u)$ for $u \in S_1(V_0)$. Then $S_1(Z_0) = \cup_{i=1}^{\infty}(N_i \cap S_1(Z_0))$.

We then only need to prove that for any i, $N_i \cap S_1(Z_0)$ is of measure zero in $\mathbb{R}^k$. We define the mapping $\tilde{\tau} : \tilde{Z}_0$ as $\tilde{\tau}(u'_2, z') = u'$ if $z' = \psi(u')$. Since Z_0 is of measure zero in $\mathbb{R}^{k_2}$, $\tilde{Z}_0$ is measure zero in $\mathbb{R}^{k_1}$. Then because $\tilde{\tau}$ is smooth, the image of $\tilde{\tau}$ of the set $\tilde{V}_0$ is of zero measure in $\mathbb{R}^{k_1}$ (The image of a zero mesure set under a smooth mapping is also measure zero). Notice that $\mathcal{B}_{\bar{\epsilon}(u)}(\bar{u}) \cap S_1(Z_0)$ is contained in the image $\tilde{\tau}(\tilde{Z}_0)$, it is also of zero measure. This finishes the proof. □

Now we prove Lemma E.1. When $l \leq L-3$, let $u^{(l)} \in \mathbb{R}^{m_l}$ be the output vector of the l-th layer. Then $u^{(l+1)} = \psi(w^{(l+1)0}, u^{(l)})$, where ψ is analytic and $w^{(l+1)0}$ is the initial parameter in the l-th layer. We now prove that $u^{(l)}$ follows a continuous distribution by induction. When $l = 0$, it is true since $u^{(0)} = (x_1, \cdots, x_n)$ is just the input data, which follows a continuous distribution under Assumption 3. Now suppose $u^{(l)}$ still follows a continuous distribution, we have: (i) since $w^{(l+1)0}$ follows a continuous distribution at the mirrored initialization, $u^{(l)}$ and $w^{(l+1)0}$ follow a continuous joint distribution; (ii) Now, viewing $(w^{(l+1)0}, u^{(l)}) \in \mathbb{R}^{m_l m_{l+1} + m_l}$ as the input of ψ, when $m_l \geq m_{l+1}$, $J(w^{(l+1)0}, u^{(l)})$ can be proved to be full row rank w.p.1. using the same technique as Theorem 1. In conclusion, we have $u^{(l+1)}$ follows a continuous distribution by Lemma E.2. Hence, we finish the proof of Lemma E.1.

F Details on experimental setup

Our empirical studies are based on the synthetic dataset, MNIST, CIFAR-10, CIFAR-100 and the R-ImageNet datasets. MNIST, CIFAR-10 and CIFAR-100 are licensed under MIT. Imagenet is licensed under Custom (non-commercial). All the experiments are run on NVIDIA V100 GPU. Here, we introduce our settings on synthetic dataset and R-ImageNet.

- Synthetic datset: For $i = 1 \ldots, 1000$, we independently generate $x_i \in \mathbb{R}^{200}$ from standard independent Gaussian, and normalize it to $\|x_i\|_2 = 1$, and we set the ground truth as $y_i = (1^T x_i)^2$ for $i = 1, \cdots, 1000$. In short, sample size $n = 1000$, input dimension $d = 200$.
- R-ImageNet: This is a specifically constructed "restricted" version of ImageNet, with resolution 224×224.

  The vanilla ImageNet dataset spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images. In our experiments, we use a subset of ImageNet, namely Restricted-ImageNet (R-ImageNet). Similar with [18], we leverage the WordNet [26] hierarchical structure of the dataset such that each class in the R-ImageNet is a superclass category composed of multiple ImageNet classes, noted in Table 2 as "components". For example, the "bird" class of R-ImageNet (both the train and validation parts) is the aggregation of ImageNet-1k classes: [10: 'brambling', 11: 'goldfinch', 12: 'house finch', 13: 'junco', 14: 'indigo bunting'], more details can be seen in Table 2. As a result, there are 20 super classes which contain a total of 190 vanilla ImageNet classes.

Table 2: Classes used in the R-ImageNet dataset. The class ranges are inclusive.

Class name	Corresponding ImageNet components
bird	$[10, 11, 12, 13, 14]$
turtle	$[33, 34, 35, 36, 37]$
lizard	$[42, 43, 44, 45, 46]$
snake	$[60, 61, 62, 63, 64]$
spider	$[72, 73, 74, 75, 76]$
crab	$[118, 119, 120, 121, 122]$
dog	$[205, 206, 207, 208, 209]$
cat	$[281, 282, 283, 284, 285]$
bigcat	$[289, 290, 291, 292, 293]$
beetle	$[302, 303, 304, 305, 306]$
butterfly	$[322, 323, 324, 325, 326]$
monkey	$[371, 372, 373, 374, 375]$
fish	$[393, 394, 395, 396, 397]$
fungus	$[992, 993, 994, 995, 996]$
musical-instrument	$[402, 420, 486, 546, 594]$
sportsball	$[429, 430, 768, 805, 890]$
car-truck	$[609, 656, 717, 734, 817]$
train	$[466, 547, 565, 820, 829]$
clothing	$[474, 617, 834, 841, 869]$
boat	$[403, 510, 554, 625, 628]$

In each dataset, the neural network architectures are chosen as follows, all of the following cases satisfy $m \geq \frac{2n}{d}$ or $m_{L-1} \geq \frac{2n}{m_{L-2}}$, where m_l is the width of the l-th layer.

- Synthetic dataset: we use 1-hidden-layer neural networks with Tanh activation (except for the last layer, where the output dimension equals 1 and no Tanh applied). We study different widths of the hidden layer among $m = 20, 40, 80, 100, 200, 400, 800, 100, 1200$. All of these cases satisfy $m \geq \frac{2n}{d}$.
- MNIST: we use 2-hidden-layer neural networks with ReLU activation (except for the last layer, where the output dimension equals the number of classes and no activation applied). The input dimension $d = 784$, the width of the 1st layer is fixed with $m_1 = 784$ and we

study different widths of the 2nd hidden layer among $m_2 = 64, 128, 256, 512, 784, 1024$. All of these cases satisfy $m_{L-1} \geq \frac{2n}{m_{L-2}}$, where m_l is the width of the l-th layer.

- CIFAR-10, CIFAR-100 and R-ImageNet: we use ResNet-18 and we try different number of channels in the 4th block among $m = 64, 128, 256, 512$ (for regular ResNet-18, the default number of channels in the 4st block should be 512). All of these cases satisfy $m_{L-1} \geq \frac{2n}{m_{L-2}}$, where m_l is the width of the l-th layer.

In each dataset, the setup for algorithms are as follows: as for training regime, we apply the mirrored LeCun's initialization for all the neural network structures mentioned above, and for regular training, we use the regular LeCun's initialization. We use square loss for the synthetic dataset, and multi-class cross entropy loss is used for the rest of the cases. During training, CIFAR-10, CIFAR-100 images are padded with 4 pixels of zeros on all sides, then randomly flipped (horizontally) and cropped. R-ImageNet images are randomly cropped during training and center-cropped during testing. Global mean and standard deviation are computed on all the training pixels and applied to normalize the inputs on each dataset. As it is required in problem (5), in our training regime, the optimization variable for and the output layer is cut off to half, i.e., $v = (v_1, \cdots, v_{\frac{m}{2}})$, the other half is always $-v$; as for the hyperparameters of $B(v)$, we set $\zeta = 0.001$ and $\kappa = 1000$. After each iteration, relevant parameters will projected in to their feasible sets. In addition to the general setup above, more customized hyperparameters are listed as follows:

- Synthetic dataset: For both our training regime and the regular training regime, $B_\epsilon(w)$ constraint is added on the weights in the hidden layer with $\epsilon = 0.1, 0.2, 0.4, 0.8, 1, 2, 4, 8, 10, 1000$ ($\epsilon = 1000$ is equivalent to the unconstrained updates for w). Gradient Descent with 0.9 momentum is used, and we use different constant learning rates lr_1, lr_2 for hidden weights and outer weights, in all cases with different m and ϵ, we grid search learning rate lr_1=[1e-4,1e-3,5e-3,1e-2,5e-2,1e-1,5e-1], lr_2=[1e-4,1e-3,5e-3,1e-2,5e-2,1e-1,5e-1] and report the best results. The neural network is trained for 200000 iterations.

- MNIST: For our training regime, $B_\epsilon(w)$ constraint is added on the weights in the 2nd layer with $\epsilon = 0.1, 0.2, 0.4, 0.8, 1, 2, 4, 8, 10, 1000$ ($\epsilon = 1000$ is equivalent to the unconstrained updates for w). In each case, we either use Adam with 0.001 initial learning rate and 1e-4 weight decay, or Stochastic Gradient Descent (SGD) with 0.01 initial learning rate, 0.9 momentum and 5e-4 weight decay, and we report the best results. For both training regime and the regular training regime, we use cosine annealing learning rate scheduling [25] with $T_{\max}$=number of epochs , and the neural network is trained for 200 epochs and batch size of 64 is used.

- CIFAR-10, CIFAR-100: For our training regime , $B_\epsilon(w^0)$ constraint is added on the 4-th block with different constraint size among $\epsilon = 0.1, 0.2, 0.4, 0.8, 1, 2, 4, 8, 10, 1000$ ($\epsilon = 1000$ is equivalent to the unconstrained updates for w). For both our training regime and the regular training regime, SGD with 0.1 initial learning rate, 0.9 momentum and 5e-4 weight decay is used, and we use cosine annealing learning rate scheduling with T_{max}=number of epochs, and the neural network is trained for 600 epochs and batch size of 128 is used.

- R-ImageNet: For our training regime, $B_\epsilon(w^0)$ constraint is added on the 4th block with different constraint size among $\epsilon = 0.01, 0.1, 1, 1000$ ($\epsilon = 1000$ is equivalent to the unconstrained updates for w), and the weights in the 4th block are projected into the constraint after each mini-batch iteration. For both our training regime and the regular training regime, SGD with 0.1 initial learning rate, 0.9 momentum and 5e-4 weight decay is used, we use a stage-wise constant learning rate scheduling with a multiplicative factor of 0.1 on epoch 30, 60 and 90. The neural network is trained for 90 epochs and batch size of 256 is used.

F.1 MNIST

Figure 7 shows the test accuracies in our training regime vs regular training regime in MNIST. With proper choice of ϵ, our training regime leads to higher test accuracies.

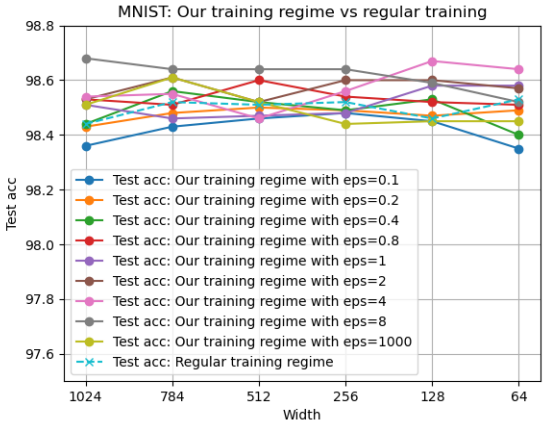
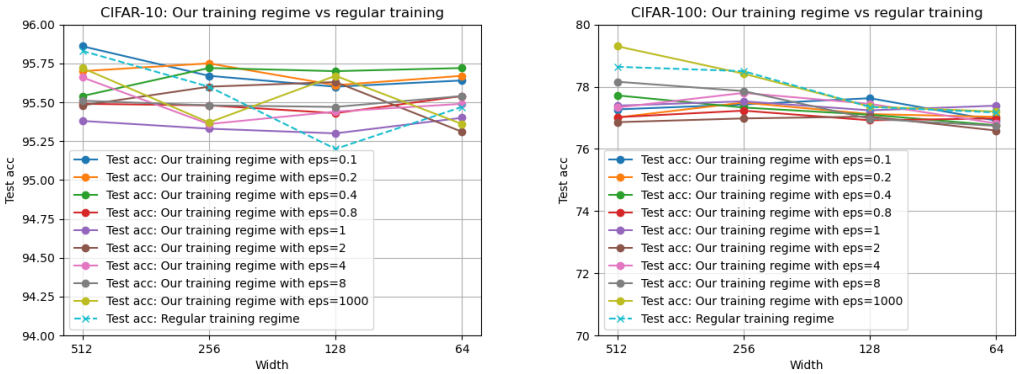


Figure 7: MNIST: test accuracies in our training regime vs regular training regime.

F.2 CIFAR-10 & CIFAR-100

In CIFAR-10 and CIFAR-100 dataset, our training regime and regular training regime have similar performance (see Figure 8). In several cases, our training regime leads to higher test accuracies. Here, regular training will not fail when we reduce the width of 4-th block, perhaps this is due to the strong expressivity of ResNet-18. In comparison, on a more complicated dataset such as R-ImageNet, narrowing ResNet-18 will jeopardize the regular training (as illustrated in Section 4 and the following subsection).



(a) CIFAR-10 (b) CIFAR-100

Figure 8: CIFAR-10 & CIFAR-100: test accuracies in our training regime vs regular training regime.

F.3 R-ImageNet

In this subsection, Figure 9 is the same figure as Figure 6 in the full paper, but with 60% confidence error bars (based on 2 seeds). Besides, Figure 10 shows a selected result of Loss & Accuracy per epoch in our training regime with $\epsilon = 0.04$ & regular training regime (here, we present the early-stopped results). As a result, our training regime can reduce the number of parameters in the 4-th block of ResNet-18 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.

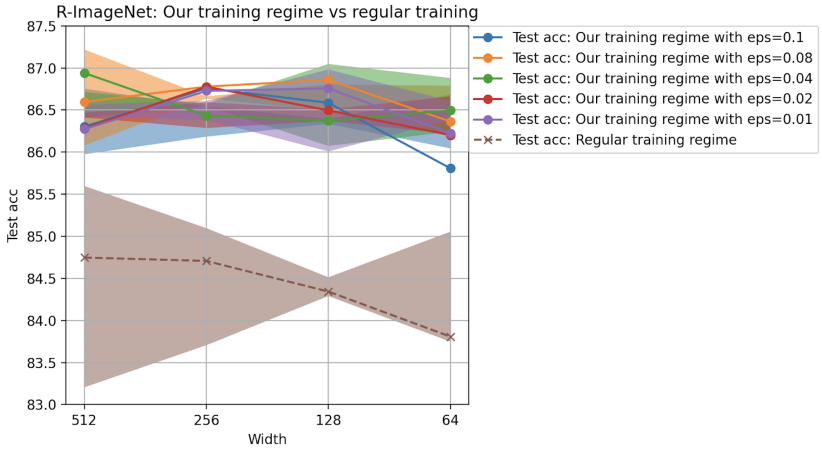
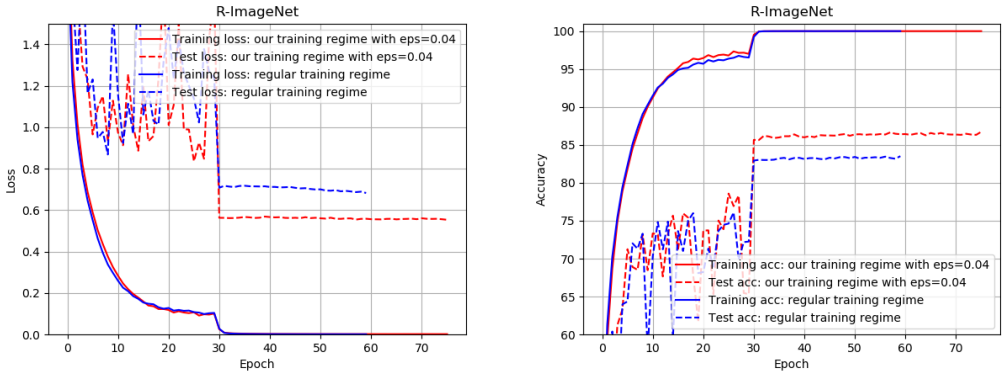


Figure 9: R-ImageNet: test accuracies under our training regime vs regular training regime. The solid & dotted lines are averaged results over 2 seeds, the shaded areas indicates the 60% confidence intervals.



(a) Loss

(b) Accuracy

Figure 10: R-ImageNet (selected): loss & Accuracy per epoch in our training regime with $\epsilon = 0.04$ & regular training regime.