Is Deeper Better only when Shallow is Good?

Eran Malach and Shai Shalev-Shwartz

School of Computer Science, The Hebrew University, Israel

Abstract

Understanding the power of depth in feed-forward neural networks is an ongoing challenge in the field of deep learning theory. While current works account for the importance of depth for the expressive power of neural-networks, it remains an open question whether these benefits are exploited during a gradient-based optimization process. In this work we explore the relation between expressivity properties of deep networks and the ability to train them efficiently using gradient-based algorithms. We give a depth separation argument for distributions with fractal structure, showing that they can be expressed efficiently by deep networks, but not with shallow ones. These distributions have a natural coarse-to-fine structure, and we show that the balance between the coarse and fine details has a crucial effect on whether the optimization process is likely to succeed. We prove that when the distribution is concentrated on the fine details, gradient-based algorithms are likely to fail. Using this result we prove that, at least in some distributions, the success of learning deep networks depends on whether the distribution can be well approximated by shallower networks, and we conjecture that this property holds in general.

1 Introduction

A fundamental question in studying deep networks is understanding why and when "deeper is better". There have been several results identifying a "depth separation" property: showing that there exist functions which are realized by deep networks of moderate width, that cannot be approximated by shallow networks, unless an exponential number of units is used. However, this is unsatisfactory, as the fact that a certain network architecture can express some function does not mean that we can learn this function from training data in a reasonable amount of training time. In fact, there is theoretical evidence showing that gradient-based algorithms can only learn a small fraction of the functions that are expressed by a given neural-network (e.g [20]).

This paper relates expressivity properties of deep networks to the ability to train them efficiently using a gradient-based algorithm. We start by giving depth separation arguments for distributions with fractal structure. In particular, we show that deep networks are able to exploit the self-similarity property of fractal distributions, and thus realize such distributions with a small number of parameters. On the other hand, we show that shallow networks need a number of parameters that grows exponentially with the intrinsic "depth" of the fractal. The advantage of fractal distributions is that they exhibit a clear coarse-to-fine structure. We show that if the distribution is more concentrated on the "coarse" details of the fractal, then even though shallower networks cannot exactly express the underlying distribution, they can still achieve a good approximation. We introduce the notion of **approximation curve**, that characterizes how the examples are distributed between the "coarse" details and the “fine” details of the fractal. The approximation curve captures the relation between the growth in the network’s depth and the improvement in approximation.

We next go beyond pure expressivity analysis, and claim that the approximation curve plays a key role not only in approximation analysis, but also in predicting the success of gradient-based optimization algorithms. Specifically, we show that if the distribution is concentrated on the “fine” details of the fractal, then gradient-based optimization algorithms are likely to fail. In other words, the “stronger” the depth separation is (in the sense that shallow networks cannot even approximate the distribution) the harder it is to learn a deep network with a gradient-based algorithm. While we prove this statement for a specific fractal distribution, we state a conjecture aiming at formalizing this statement in a more general sense. Namely, we conjecture that a distribution which cannot be approximated by a shallow network cannot be learned using gradient-based algorithm, even when using a deep architecture. We perform experiments on learning fractal distributions with deep networks trained with SGD and assert that the approximation curve has a crucial effect on whether a depth efficiency is observed or not. These results provide new insights as to when such deep distributions can be learned.

Admittedly, this paper is focused on analyzing a family of distributions that is synthetic by nature. That said, we note that the conclusions from this analysis may be interesting for the broader effort of understanding the power of depth in neural-networks. As mentioned, we show that there exist distributions with depth separation property (that are expressed efficiently with deep networks but not with shallow ones), that cannot be learned by gradient-based optimization algorithms. This result implies that any depth separation argument that does not consider the optimization process should be taken with a grain of salt. Additionally, our results hint that the success of learning deep networks depends on whether the distribution can be approximated by shallower networks. Indeed, this property is often observed in real-world distributions, where deeper networks perform better, but shallower networks exhibit good (if not perfect) performance.

2 Related Work

In recent years there has been a large number of works studying the expressive power of deep and shallow networks. The main goal of this research direction is to show families of functions or distributions that are realizable with deep networks of modest width, but require exponential number of neurons to approximate by shallow networks. We refer to such results as depth separation results.

Many of these works consider various measures of “complexity” that grow exponentially fast with the depth of the network, but not with the width. Hence, such measures provide a clear separation between deep and shallow networks. For example, the works of [13, 12, 11, 18] show that the number of linear regions grows exponentially with the depth of the network, but only polynomially with the width. The work of [15] shows that for networks with random weights, the curvature of the functions calculated by the networks grows exponentially with depth but not with width. In another work, [16] shows that the trajectory length, which measures the change in the output along a one-dimensional path, grows exponentially with depth. Finally, the work of [22] utilizes the number of oscillations in the function to give a depth separation result.

While such works give general characteristics of function families, they take a seemingly worst-case approach. Namely, these works show that there exist functions implemented by deep networks that are hard to approximate with a shallow net. But as in any worst-case analysis, it is not clear whether such analysis applies to the typical cases encountered in the practice of neural-networks.

In order to answer this concern, recent works show depth separation results for narrower families of functions that appear simple or "natural". For example, the work of [21] shows a very simple construction of a function on the real line that exhibits a depth separation property. The works of [7, 17] show a depth separation argument for very natural functions, like the indicator function of the unit ball. The work of [5] gives similar results for a richer family of functions. Another series of works by [10, 14] show that compositional functions, namely functions of functions, can be well approximated by deep networks. The works of [6, 4] show that compositional properties establish depth separation for sum-product networks.

Our work shares similar motivations with the above works. Namely, our goal is to construct a family of distributions that demonstrate the power of deep networks over shallow ones. Unlike these works, we do not limit ourselves to expressivity results alone, but rather take another step into exploring whether these distributions can be learned by gradient-based optimization algorithms.

Finally, while we are not aware of any work that directly considers fractal structures in the context of deep learning, there are a few works that tie them to other fields in machine learning. Notably, [2] gives a thorough review of fractal geometry from a machine learning perspective, suggesting that their structure can be exploited in various machine learning tasks. The work of [9] considers the effect of fractal structure on the performance of nearest neighbors algorithms. We also note that fractal structures are exploited in image compression (refer to [1] for a review). These works mainly give motivation to look at fractal geometry in the context of deep learning, as these seem relevant for other problems in machine learning.

3 Preliminaries

Let $\mathcal{X} = \mathbb{R}^d$ be the domain space and $\mathcal{Y} = \{\pm 1\}$ be the label space. We consider distributions defined over sets generated by an iterated function system (IFS). An IFS is a method for constructing fractals, where a finite set of contraction mappings are applied iteratively, starting with some arbitrary initial set. Applying such process ad infinitum generates a self-similar fractal. In this work we will consider sets generated by performing a finite number of iterations from such process. We refer to the number of iterations of the IFS as the "depth" of the generated set.

Formally, an IFS is defined by a set of r contractive affine [1] transformations $F = (F_1, \dots, F_r)$, where $F_i(\boldsymbol{x}) = \boldsymbol{M}^{(i)}\boldsymbol{x} + \boldsymbol{v}^{(i)}$ with full-rank matrix $\boldsymbol{M}^{(i)} \in \mathbb{R}^{d \times d}$, vector $\boldsymbol{v}^{(i)} \in \mathbb{R}^d$, s.t $\|F_i(\boldsymbol{x}) - F_i(\boldsymbol{y})\| < \|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ (we use $\|\cdot\|$ to denote the ℓ_2 norm, unless stated otherwise). We define the set $K_n \subseteq \mathcal{X}$ recursively by:

- $K_0 = [-1, 1]^d$
- $K_n = F_1(K_{n-1}) \cup \dots \cup F_r(K_{n-1})$

The IFS construction is shown in figure 1.

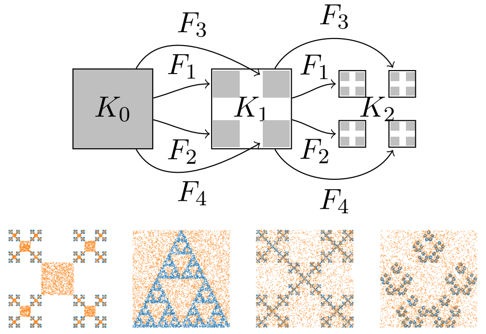

Figure 1: IFS and fractal distributions.

We define a "fractal distributions", denoted $\mathcal{D}_n$, to be any balanced distribution over $\mathcal{X} \times \mathcal{Y}$ such that positive examples are sampled from the set K_n and negative examples are sampled from

[1]In general, IFSs can be constructed with non-linear transformations, but in this paper we discuss only affine IFS.

its complement. Formally, $\mathcal{D}_n = \frac{1}{2}(\mathcal{D}_n^+ + \mathcal{D}_n^-)$ where $\mathcal{D}_n^+$ is a distribution over $\mathcal{X} \times \mathcal{Y}$ that is supported on $K_n \times \{+1\}$, and $\mathcal{D}_n^-$ is a distribution over $\mathcal{X} \times \mathcal{Y}$ that is supported on $(\mathcal{X} \setminus K_n) \times \{-1\}$. Examples for such distributions are given in figure 1 and figure 2.

In this paper we consider the problem of learning fractal distributions with feed-forward neural-networks equipped with the ReLU activation. A ReLU neural-network $\mathcal{N}_{\mathbf{W},\boldsymbol{B}} : \mathcal{X} \to \mathcal{Y}$ of depth t and width k is a function defined recursively such that $\boldsymbol{x}^{(t)} := \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})$, and:

1. $\boldsymbol{x}^{(0)} = \boldsymbol{x}$
2. $\boldsymbol{x}^{(t')} = \sigma(\boldsymbol{W}^{(t')}\boldsymbol{x}^{(t'-1)} + \boldsymbol{b}^{(t')})$ for every $t' \in \{1, \dots, t-1\}$
3. $\boldsymbol{x}^{(t)} = \boldsymbol{W}^{(t)}\boldsymbol{x}^{(t-1)} + \boldsymbol{b}^{(t)}$

Where $\boldsymbol{W}^{(1)} \in \mathbb{R}^{k \times d}, \boldsymbol{W}^{(2)}, \dots, \boldsymbol{W}^{(t-1)} \in \mathbb{R}^{k \times k}, \boldsymbol{W}^{(t)} \in \mathbb{R}^{1 \times k}, \boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(t-1)} \in \mathbb{R}^k, \boldsymbol{b}^{(t)} \in \mathbb{R}$, and $\sigma(\boldsymbol{x}) := \max(\boldsymbol{x}, 0)$.

We denote by $\mathcal{H}_{k,t}$ the family of all functions that are implemented by a neural-network of width k and depth t. Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, we denote the error of a network $h \in \mathcal{H}_{k,t}$ on distribution $\mathcal{D}$ to be $L_{\mathcal{D}}(h) := \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}[\text{sign}(h(\boldsymbol{x})) \neq y]$. We denote the approximation error of $\mathcal{H}_{k,t}$ on $\mathcal{D}$ to be the minimal error of any such function: $L_{\mathcal{D}}(\mathcal{H}_{k,t}) := \min_{h \in \mathcal{H}_{k,t}} L_{\mathcal{D}}(h)$.

4 Expressivity and Approximation

In this section we analyze the expressive power of deep and shallow neural-networks w.r.t fractal distributions. We show two results. The first is a depth separation property of neural-networks. Namely, we show that shallow networks need an exponential number of neurons to realize such distributions, while deep networks need only a number of neurons that is linear in the problem's parameters. The second result bounds the approximation error achieved by networks that are not deep enough to achieve zero error. This bound depends on the specific properties of the fractal distribution.

We analyze IFSs where the images of the initial set K_0 under the different mappings do not overlap. This property allows the neural-network to "reverse" the process that generates the fractal structure. Additionally, we assume that the images of K_0 (and therefore the entire fractal), are contained in K_0, which means that the fractal does not grow in size. This is a technical requirement that could be achieved by correctly scaling the fractal at each step. While these requirements are not generally assumed in the context of IFSs, they hold for many common fractals (cantor set, sierpinsky triangle and more). Formally, we assume the following:

Assumption 1 *There exists $\epsilon > 0$ such that for $i \neq j \in [r]$ it holds that $d(F_i(K_0), F_j(K_0)) > \epsilon$, where $d(A, B) = \min_{\boldsymbol{x} \in A, \boldsymbol{y} \in B} \|\boldsymbol{x} - \boldsymbol{y}\|$.*

Assumption 2 *For each $i \in [r]$ it holds that $F_i(K_0) \subseteq K_0$.*

Finally, as in many other problems in machine learning, we assume the positive and negative examples are separated by some margin. Specifically, we assume that the positive examples are sampled from strictly inside the set K_n, with margin γ from the set boundary. Formally, for some set A, we define A^γ to be the set of all points that are far from the boundary of A by at least γ: $A^\gamma := \{\boldsymbol{x} \in A \ : \ B_\gamma(\boldsymbol{x}) \subseteq A\}$, where $B_\gamma(\boldsymbol{x})$ denotes a ball around $\boldsymbol{x}$ with radius γ. So our assumption is the following:

Assumption 3 *There exists* $\gamma > 0$ *such that* $\mathcal{D}_n^+$ *is supported on* $K_n^\gamma \times \{+1\}$.

4.1 Depth Separation

We show that neural-networks with depth linear in n (where n is the "depth" of the fractal) can achieve zero error on any fractal distribution satisfying the above assumptions, with only linear width. On the other hand, a shallow network needs a width exponential in n to achieve zero error on such distributions.

To separate such fractal distributions, we start with the following:

Theorem 1 *There exists a neural-network of width* $5dr$ *and depth* $2n+1$, *s.t* $\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(K_n^\gamma)) = 1$ *and* $\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\mathcal{X} \setminus K_n)) = -1$.

Since, by assumption, there are no examples in the margin area, we immediately get an expressivity result under any fractal distribution:

Corollary 1 *For any distribution* $\mathcal{D}_n$ *there exist neural-network of width* $5dr$ *and depth* $2n+1$, *such that* $L_{\mathcal{D}_n}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) = 0$.

We defer the proof of Theorem 1 to the appendix, and give here an intuition of how deep networks can express these seemingly complex distributions with a small number of parameters. Note that by definition, the set K_n is composed of r copies of the set K_{n-1}, mapped by different affine transformations. In our construction, each block of the network folds the different copies of K_{n-1} on-top of each other, while "throwing away" the rest of the examples (by mapping them to a distinct value). The next block can then perform the same thing on all copies of K_{n-1} together, instead of decomposing each subset separately. This allows a very efficient utilization of the network parameters.

The above results show that deep networks are very efficient in utilizing the parameters of the network, requiring a number of parameters that grows linearly with r, d and n. Now, we want to consider the case of shallower networks, when the depth is not large enough to achieve zero error with linear width. Specifically, we show that when decreasing the depth of the network by a factor of s, we can achieve zero error by allowing the width to grow like r^s.

Notice that for any s that divides n, any IFS of depth n with r transformations can be written as depth $\frac{n}{s}$ IFS with r^s transformations. Indeed, for $\boldsymbol{i} = (i_1, \dots, i_s) \in [r]^s$ denote $F_{\boldsymbol{i}}(\boldsymbol{x}) = F_{i_1} \circ \dots \circ F_{i_s}(\boldsymbol{x})$, and we have: $K_s = \cup_{\boldsymbol{i} \in [r]^s} F_{\boldsymbol{i}}(K_0)$. So we can write a new IFS with transformations $\{F_{\boldsymbol{i}}\}_{\boldsymbol{i} \in [r]^s}$, and these will generate K_n in $\frac{n}{s}$ iterations. This gives us a stronger version of the previous result, which explicitly shows the trade-off between linear growth of depth and exponential growth of width:

Corollary 2 *For any distribution* $\mathcal{D}_n$ *and every natural* $s \leq n$ *there exists a neural-network of width* $5dr^s$ *and depth* $2\lfloor n/s \rfloor + 2$, *such that* $L_{\mathcal{D}_n}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) = 0$.

This is an upper bound on the required width of a network that can realize $\mathcal{D}_n$, for any given depth. To show the depth separation property, we show that a shallow network needs an exponential number of neurons to implement the indicator function. This gives the equivalent lower bound on the required width.

Theorem 2 *Let $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ be a network of depth t and of width k, such that* $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(K_n^\gamma)) = 1$ *and* $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\mathcal{X} \setminus K_n)) = -1$*. Denote s to be the ratio between the depth of the fractal and the depth of the network, so $s := n/t$. Then the width of the network grows exponentially with s, namely: $k \geq \frac{d}{e} r^{s/d}$.*

Proof From Proposition 3 in [11] we get that there are $\prod_{t'=1}^{t} \sum_{j=0}^{d} \binom{k}{j} \leq (ek/d)^{td}$ linear regions in $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ (where we use Lemma A.5 from [19]). Furthermore, every such linear region is an intersection of affine half-spaces.

Note that any function such that $\operatorname{sign}(f(K_n^\gamma)) = 1$ and $\operatorname{sign}(f(\mathcal{X} \setminus K_n)) = -1$ has at least r^n such linear regions. Indeed, notice that $K_n = \cup_{\boldsymbol{i} \in [r]^n} F_{\boldsymbol{i}}(K_0)$. Assume by contradiction that there are $< r^n$ linear regions, so there exists $\boldsymbol{i} \neq \boldsymbol{j} \in [r]^n$ such that $F_{\boldsymbol{i}}(K_0), F_{\boldsymbol{j}}(K_0)$ are in the same linear region. Fix $\boldsymbol{x} \in F_{\boldsymbol{i}}(K_0)^\gamma, \boldsymbol{y} \in F_{\boldsymbol{j}}(K_0)^\gamma$ and observe the function f along the line from $\boldsymbol{x}$ to $\boldsymbol{y}$. By our assumption $f(\boldsymbol{x}) \geq 0$, $f(\boldsymbol{y}) \geq 0$. This line must cross $\mathcal{X} \setminus K_n$, since from Assumption 1 we get that $d(F_{\boldsymbol{i}}(K_0), F_{\boldsymbol{j}}(K_0)) > 0$ for every $\boldsymbol{i} \neq \boldsymbol{j} \in [r]^n$. Therefore f must get negative values along the line between $\boldsymbol{x}$ to $\boldsymbol{y}$, so it must cross zero at least twice. Every linear region is an intersection of half-spaces, and hence convex, so f is linear on this path, and we reach a contradiction.

Therefore, we get that $(ek/d)^{td} \geq r^n$, and therefore: $k \geq \frac{d}{e} r^{s/d}$. ■

This result implies that there are many fractal distributions for which a shallow neural-network needs exponentially many neurons to achieve zero error on. In fact, we show this for any distribution without "holes" (areas of non-zero volume with no examples from $\mathcal{D}_n$, outside the margin area):

Corollary 3 *Let $\mathcal{D}_n$ be some fractal distribution, s.t for every ball $B \subseteq K_n^\gamma \cup (\mathcal{X} \setminus K_n)$ it holds that $\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\boldsymbol{x} \in B\right] > 0$. Then for every depth t and width k, s.t $k < \frac{d}{e} r^{\frac{n}{td}}$, we have $L_{\mathcal{D}_n}(\mathcal{H}_{k,t}) > 0$.*

Proof Let $\mathcal{N}_{\mathbf{W},\boldsymbol{B}} \in \mathcal{H}_{k,t}$. From Theorem 2 there exists $\boldsymbol{x} \in K_n^\gamma$ with $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})) = -1$ or otherwise there exists $\boldsymbol{x} \in \mathcal{X} \setminus K_n$ with $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})) = 1$. Assume w.l.o.g that we have $\boldsymbol{x} \in K_n^\gamma$ with $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})) = -1$. Since $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ is continuous, there exists a ball around $\boldsymbol{x}$, with $\boldsymbol{x} \in B \subseteq K_n^\gamma$, such that $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(B)) = -1$. From the properties of the distribution we get:

$$\begin{aligned}\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})) \neq y\right] &\geq \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})) \neq y \textit{ and } \boldsymbol{x} \in B\right] \\ &= \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\boldsymbol{x} \in B\right] > 0\end{aligned}$$

■

The previous result shows that in many cases we cannot guarantee exact realization of "deep" distributions by shallow networks that are not exponentially wide. On the other hand, we show that in some cases we may be able to give good guarantees on *approximating* such distributions with shallow networks, when we take into account how the examples are distributed within the fractal structure. We will formalize this notion in the next part of this section.

4.2 Approximation Curve

Given distribution $\mathcal{D}_n$, we define the **approximation curve** of this distribution to be the function $P : [n] \to [0, 1]$, where:

$$P(j) = \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\boldsymbol{x} \notin K_j \textit{ or } y = 1\right]$$

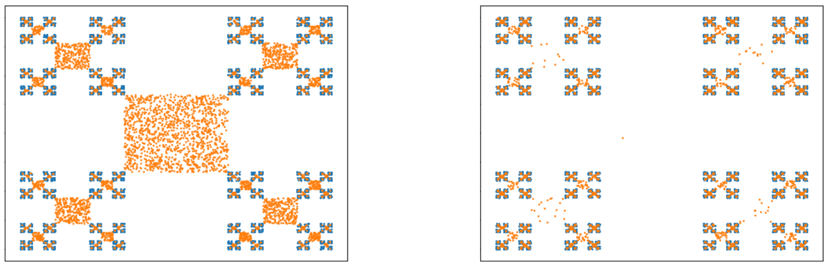

Figure 2: 2D cantor distributions of depth 5, negative examples in orange and positive in blue. The negative examples are concentrated in the middle rectangle, and not in all $\mathcal{X} \setminus K_n$. Left: "coarse" approximation curve (curve#1). Right: "fine" approximation curve (curve#4).

Notice that $P(0) = \frac{1}{2}$, $P(n) = 1$, and that P is non-decreasing. The approximation curve P captures exactly how the negative examples are distributed between the different levels of the fractal structure. If P grows fast at the beginning, then the distribution is more concentrated on the low levels of the fractal (coarse details). If P stays flat until the end, then most of the weight is on the high levels (fine details). Figure 2 shows samples from two distributions over the same fractal structure, with different approximation curves.

A simple argument shows that distributions concentrated on coarse details can be well approximated by shallower networks. The following theorem characterizes the relation between the approximation curve and the "approximability" by networks of growing depth:

Theorem 3 *Let $\mathcal{D}_n$ be some fractal distribution with approximation curve P. Fix some j, s, then for $\mathcal{H}_{k,t}$ with depth $t = 2\lfloor j/s \rfloor + 2$ and width $k = 5dr^s$, we have: $L_{\mathcal{D}_n}(\mathcal{H}_{k,t}) \leq 1 - P(j)$.*

Proof From Theorem 1 and Corollary 2, there exists a network of depth $t = 2\lfloor j/s \rfloor + 2$ and width $5dr^s$ such that $\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(K_j^\gamma)) = 1$ and $\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\mathcal{X} \setminus K_j)) = -1$. Notice that since $K_n^\gamma \subseteq K_j^\gamma$, we have: $\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\boldsymbol{x} \notin K_j^\gamma \text{ and } y = 1\right] = 0$. Therefore for this network we get: $\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x})) \neq y\right] \leq \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}_n}\left[x \in K_j \text{ and } y \neq 1\right] = 1 - P(j)$. ■

This shows that using the approximation curve of distribution $\mathcal{D}_n$ allows us to give an upper bound on the approximation error for networks that are not deep enough. We give a lower bound for this error in a more restricted case. We limit ourselves to the case where $d = 1$, and observe networks of width $k < r^s$ for some s. Furthermore, we assume that the probability of seeing each subset of the fractal is the same. Then we get the following theorem:

Theorem 4 *Assume that $\mathcal{D}_n$ is a distribution on $\mathbb{R}$ ($d = 1$). Note that for every j, K_j is a union of r^j intervals, and we denote $K_j = \cup_{i=1}^{r^j} I_i$ for intervals I_i. Assume that the distribution over each interval is equal, so for every i, ℓ, y': $\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in I_i \text{ and } y = y'\right] = \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in I_\ell \text{ and } y = y'\right]$. Then for depth t and width $k < r^s$, for $n > j > st$ we get: $L_{\mathcal{D}_n}(\mathcal{H}_{k,t}) \geq (1 - r^{st-j})(1 - P(j))$.*

The above theorem shows that for shallow networks, for which $st \ll j$, the approximation curve gives a very tight lower bound on the approximation error. This is due to the fact that shallow networks have a limited number of linear regions, and hence effectively give constant prediction on most of the "finer" details of the fractal distribution. This result implies that there are fractal distributions that are not only hard to realize by shallow networks, but that are even hard to approximate. Indeed, fix some small $\epsilon > 0$ and let $j := st + \log_r(\frac{1}{2\epsilon})$. Then if the approximation curve stays flat for the first j levels (i.e $P(j) = \frac{1}{2}$), then from Theorem 4 the approximation error is at least $\frac{1}{2} - \epsilon$.

This gives a **strong depth separation** result: shallow networks have an error of $\approx \frac{1}{2}$ while a network of depth $t \geq 2\lfloor n/s \rfloor + 2$ can achieve zero error (on any fractal distribution). This strong depth separation result occurs when the distribution is concentrated on the "fine" details, i.e when the approximation curve stays flat throughout the "coarse" levels. In the next section we relate the approximation curve to the success of fitting a deep network to the fractal distribution, using gradient-based optimization algorithms. Specifically, we claim that distributions with strong depth separation **cannot** be learned by any network, deep or shallow, using gradient-based algorithms.

5 Optimization Analysis

So far, we analyzed the ability of neural-networks to express and approximate different fractal distributions. But it remains unclear whether these networks can be learned with gradient-based optimization algorithms. In this section, we show that the success of the optimization highly depends on the approximation curve of the fractal distribution. Namely, we show that for distributions with a "fine" approximation curve, that are concentrated on the "fine" details of the fractal, the optimization fails with high probability, for **any** gradient-based optimization algorithm.

To simplify the analysis, we focus in this section on a very simple fractal distribution: a distribution over the Cantor set in $\mathbb{R}$. We begin by defining the standard construction of the Cantor set, using an IFS. We construct the set C_n recursively:

1. $C_0 = [0, 1]$
2. $C_n = F_1(C_{n-1}) \cup F_2(C_{n-1})$

where $F_1(x) = \frac{1}{3} - \frac{1}{3}x$ and $F_2(x) = \frac{1}{3} + \frac{1}{3}x$.

Now, fix margin $\gamma < \frac{3^{-n}}{2}$. We define the distribution $\mathcal{D}_n^+$ to be the uniform distribution over $C_n^\gamma \times \{+1\}$. The distribution $\mathcal{D}_n^-$ is a distribution over $C_0 \setminus C_n$, where we sample from each "level" C_j ($j < n$) with probability p_j. Formally, we define $E_j := C_{j-1} \setminus C_j$ to be the j-th level of the negative distribution. We use $\mathcal{U}(E_j)$ to denote the uniform distribution on set E_j, then: $\mathcal{D}_n^- = \sum_{j=1}^n p_j \left(\mathcal{U}(E_j) \times \{-1\}\right)$. Notice that the approximation curve of this distribution is given by: $P(j) = \frac{1}{2} + \frac{1}{2}\sum_{i=1}^j p_i$. As before, we wish to learn $\mathcal{D}_n = \frac{1}{2}(\mathcal{D}_n^+ + \mathcal{D}_n^-)$. Figure 3 shows a construction of such distribution.

5.1 Hardness of Optimization

The main theorem in this section shows the connection between the approximation curve and the behavior of a gradient-based optimization algorithm. This result shows that for deep enough cantor distributions, the value of the approximation curve on the fine details of the fractal bounds the norm of the population gradient for randomly initialized network:

Theorem 5 *Fix some depth t, width k and some $\delta \in (0, 1)$. Let $n, n' \in \mathbb{N}$ such that $n > n' > \log^{-1}(\frac{3}{2}) \log(\frac{4tk^2}{\delta})$. Let $\mathcal{D}_n$ be some cantor distribution with approximation curve P. Assume* *we initialize a neural-network $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ of depth t and width k, with weights initialized uniformly in $[-\frac{1}{2n_{in}}, \frac{1}{2n_{in}}]$ (where n_{in} denotes the in-degree of each neuron), and biases initialized with a fixed value $b = \frac{1}{2}$* [2]*. Denote the hinge-loss of the network on the population by:*

$$\mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) = \mathbb{E}_{(x,y)\sim\mathcal{D}^n}\left[\max\{1 - y\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x), 0\}\right]$$

Then with probability at least $1-\delta$ we have:

1. $\left\|\frac{\partial}{\partial \mathbf{W}}\mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}})\right\|_{\max} \le 5\left(P(n') - \frac{1}{2}\right)$
   $\left\|\frac{\partial}{\partial \boldsymbol{B}}\mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}})\right\|_{\max} \le 3\left(P(n') - \frac{1}{2}\right)$
2. $L_{\mathcal{D}_n}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) \ge \left(\frac{3}{2} - P(n')\right)(1 - P(n'))$

Where we denote $\|\mathbf{A}\|_{\max} = \max|a_{i_1,\dots,i_j}|$ for some tensor $\mathbf{A}$.

We give the full proof of the theorem in the appendix, and show a sketch of the argument here. Observe the distribution $\mathcal{D}_n$, limited to the set $C_{n'}$. Notice that the volume of $C_{n'}$ (namely, the sum of the lengths of its intervals) decreases exponentially fast with n'. Since each neuron of the network corresponds to a separation of the space by a hyper-plane, we get that the probability of each hyper-plane to separate an interval of $C_{n'}$ decreases exponentially fast with n'. Thus, for n' that is logarithmic in the number of neurons, there is a high probability that each interval of $C_{n'}$ is not separated by any neuron. In this case the network is linear on each interval. A simple argument gives a bound on the gradient of a linear classifier on each interval, and on its classification error. Note that the approximation curve determines how much of the distribution is concentrated on the set $C_{n'}$. That is, if $P(n') - \frac{1}{2}$ is close to zero, this means that most of the distribution is concentrated on $C_{n'}$. Using this property allows us to bound the norm of the gradient in terms of the approximation curve.

We now give some important implications of this theorem. First, notice that we can define cantor distributions for which a gradient-based algorithm fails with high probability. Indeed, we define the "fine" cantor distribution to be a distribution concentrated on the highest level of the cantor set. Given our previous definition, this means $p_1, \dots, p_{n-1} = 0$ and $p_n = 1$. The approximation curve for this distribution is therefore $P(0) = \cdots = P(n-1) = \frac{1}{2}$, $P(n) = 1$. Figure 3 shows the "fine" cantor distribution drawn over its composing intervals. From Theorem 5 we get that for $n > \log^{-1}(\frac{3}{2})\log(\frac{4tk^2}{\delta})$, with probability at least $1-\delta$, the population gradient is zero and the error is $\frac{1}{2}$. This result immediately implies that vanilla gradient-descent on the distribution will be stuck in the first step. But SGD, or GD on a finite sample, may move from the initial point, due to the stochasticity of the gradient estimation. What the theorem shows is that the objective is extremely flat almost everywhere in the regime of $\mathbf{W}$, so stochastic gradient steps are highly unlikely to converge to any solution with error better than $\frac{1}{2}$.

The above argument shows that there are fractal distributions that can be **realized** by deep networks, for which a standard optimization process is likely to fail. We note that this result is interesting by itself, in the broader context of depth separation results. It implies that for many deep architectures, there are distributions with depth separation property that cannot be learned by standard optimization algorithms:

[2]We note that it is standard practice to initialize the bias to a fixed value. We fix $b = \frac{1}{2}$ for simplicity, but a similar result can be given for any choice of $b \in \left[0, \frac{1}{2}\right]$.

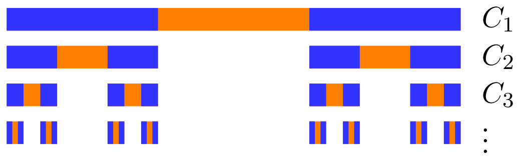

Figure 3: "Fine" cantor distributions of growing depth. Negative areas in orange, positive in blue.

Corollary 4 *There exist two constants c_1, c_2, such that for every width $k \geq 10$ and $\delta \in (0,1)$, for every depth $t > c_1 \log(\frac{k}{\delta}) + c_2$ there exists a distribution $\mathcal{D}$ on $\mathbb{R} \times \{\pm 1\}$ for which:*

1. *$\mathcal{D}$ can be realized by a neural network of depth t and width 10.*
2. *$\mathcal{D}$ cannot be realized by a one-hidden layer network with less than 2^{t-1} units.*
3. *Any gradient-based algorithm trying to learn a neural-network of depth t and width k, with initialization and loss described in Theorem 5, returns a network with error $\frac{1}{2}$ w.p $\geq 1 - \delta$.*

We can go further, and use Theorem 5 to give a better characterization of these hard distributions. Recall that in the previous section we showed distributions that exhibit a *strong* depth separation property: distributions that are realizable by deep networks, for which shallow networks get an error exponentially close to $\frac{1}{2}$. From Theorem 5 we get that any cantor distribution that gives a strong depth separation **cannot** be learned by gradient-based algorithms:

Corollary 5 *Fix some depth t, width k and some $\delta \in (0,1)$. Let $n > 4 \log^{-1}(\frac{3}{2}) \log(\frac{4tk^2}{\delta}) + 2$. Let $\mathcal{D}_n$ be some cantor distribution such that any network of width 10 and depth $t' < n$ has an error of at least $\frac{1}{2} - \epsilon^{n-t'}$, for some $\epsilon \in (0,1)$ (i.e, strong depth separation). Assume we initialize a network of depth t and width k as described in Theorem 5. Then with probability at least $1 - \delta$:*

1. $\left\| \frac{\partial}{\partial \mathbf{W}} \mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) \right\|_{\max} \leq 5\epsilon^{n/2}$
   $\left\| \frac{\partial}{\partial \boldsymbol{B}} \mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) \right\|_{\max} \leq 3\epsilon^{n/2}$
2. $L_{\mathcal{D}_n}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}) \geq \frac{1}{2} - \frac{3}{2}\epsilon^{n/2}$

Proof Using Theorem 3 and the strong depth separation property we get that for every t' we have: $1 - P\left(\frac{t'-1}{2}\right) \geq L_{\mathcal{D}}(\mathcal{H}_{10,t'}) \geq \frac{1}{2} - \epsilon^{n-t'}$. Choosing $t' = \frac{n}{2}$ and taking $n' = \frac{n}{4} - \frac{1}{2}$ we get $P(n') \leq \frac{1}{2} + \epsilon^{n/2}$. By the choice of n we can apply Theorem 5 and get the required. ■

This shows that in the strong depth separation case, the population gradient is exponentially close to zero with high probability. Effectively, this property means that even a small amount of stochastic noise in the gradient estimation (for example, in SGD), makes the algorithm fail.

This result gives a very important property of cantor distributions. It shows that every cantor distribution that cannot be **approximated** by a **shallow** network (achieving error greater than $\frac{1}{2}$), cannot be **learned** by a **deep** network (when training with gradient-based algorithms). While we show this in a very restricted case, we conjecture that this property holds in general:

Conjecture 1 *Let $\mathcal{D}$ be some distribution such that $L_{\mathcal{D}}(\mathcal{H}_{k,t}) = 0$ (realizable with networks of width k and depth t). If $L_{\mathcal{D}}(\mathcal{H}_{k,t'})$ is exponentially close to $\frac{1}{2}$ when $t' \to 1$, then any gradient-based algorithm training a network of depth t and width k will fail with high probability.*

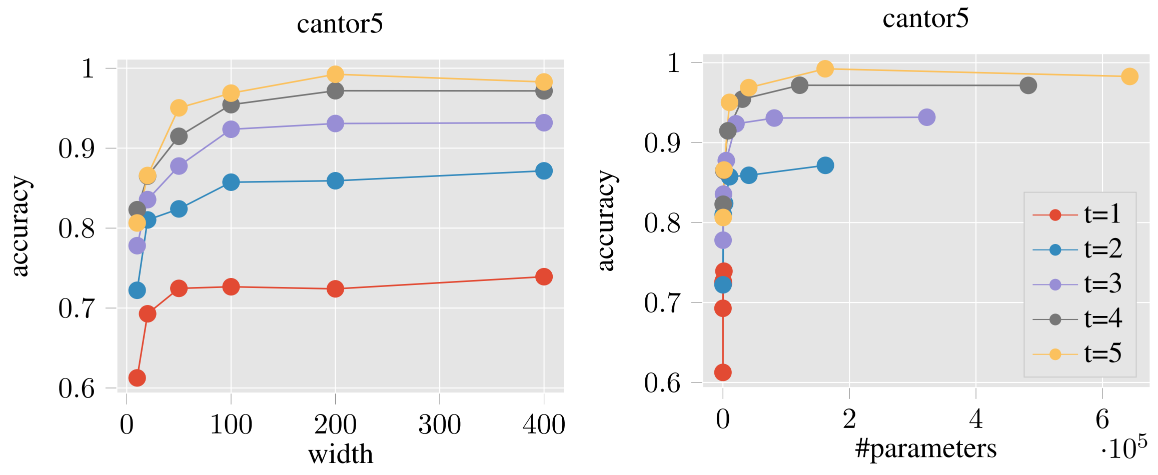

Figure 4: The effect of depth on learning the cantor set.

6 Experiments

In the previous section, we saw that learning a "fine" distribution with gradient-based algorithms is likely to fail. To complete the picture, we now show a positive result, asserting that when the distribution has enough weight on the "coarse" details, SGD succeeds to learn a deep network with small error. Moreover, we show that when training on such distributions, a clear depth separation is observed, and deeper networks indeed perform better than shallow networks. Unfortunately, giving theoretical evidence to support this claim seems out of reach, as analyzing gradient-descent on deep networks proves to be extremely hard due to the dynamics of the non-convex optimization. Instead, we perform experiments to show these desired properties.

In this section we present our experimental results on learning deep networks with Adam optimizer ([8]), trained on samples from fractal distributions. First, we show that depth separation is observed when training on fractal distributions with "coarse" approximation curve: deeper networks perform better and have better parameter utilization. Second, we demonstrate the effect of training on "coarse" vs. "fine" distributions, showing that the performance of the network degrades as the approximation curve becomes finer.

We start by observing a distribution with a "coarse" approximation curve (denoted curve #1), where the negative examples are evenly distributed between the levels. The underlying fractal structure is a two-dimensional variant of the cantor set. This set is constructed by an IFS with four mappings, each one maps the structure to a rectangle in a different corner of the space. The negative examples are concentrated in the central rectangle of each structure. The distributions are shown in figure 2.

We train feed-forward networks of varying depth and width on a 2D cantor distribution of depth 5. We sample 50K examples for a train dataset and 5K examples for a test dataset. We train the networks on this dataset with Adam optimizer for 10^6 iterations, with batch size of 100 and different learning rates. We observe the best performance of each configuration (depth and width) on the test data along the runs. The results of these experiments are shown in figure 4.

In this experiment, we see that a wide enough depth 5 network gets almost zero error. The fact that the network needs much more parameters than the best possible network is not surprising, as

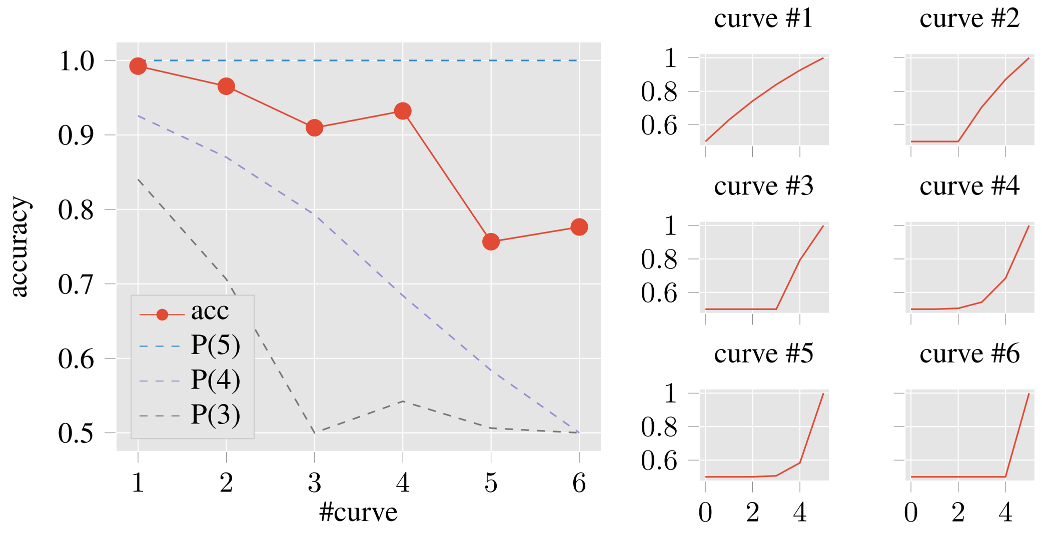

Figure 5: Learning depth 5 network on 2D cantor set of depth 5, with different approximation curves. The figures show the values of the approximation curve (denoted P) at different levels of the fractal. Large values correspond to more weight. In red is the accuracy of the best depth 5 network architecture trained on these distributions.

previous results have shown that over-parametrization is essential for the optimization to succeed ([3]). Importantly, we can see a clear depth separation between the networks: deeper networks achieve better accuracy, and are more efficient in utilizing the network parameters.

Next, we observe the effect of the approximation curve on learning the distribution. We compare the performance of the best depth 5 networks, when trained on distributions with different approximation curves. The training and validation process is as described previously. We also plot the value of the approximation curve for each distribution, in levels $3, 4, 5$ of the fractal. The results of this experiment are shown in figure 5. Clearly, the approximation curve has a crucial effect on learning the distribution. While for "coarse" approximation curves the network achieves an error that is close to zero, we can see that distributions with "fine" approximation curves result in a drastic degradation in performance.

We perform the same experiments with different fractal structures (figure 1 at the beginning of the paper shows an illustration of the distributions we use). Tables 1, 2 in the appendix summarize the results of all the experiments. We note that the effect of depth can be seen clearly in all fractal structures. The effect of the approximation curve is observed in the Cantor set and the Pentaflake and Vicsek sets (fractals generated by an IFS with 5 transformations). In the Sierpinsky Triangle (generated with 3 transformations), the approximation curve seems to have no effect when the width of the network is large enough. This might be due to the fact that a depth 5 IFS with 3 transformations generates a relatively small number of linear regions, making the problem overall relatively easy, even when the underlying distribution is hard.

Acknowledgements: This research is supported by the European Research Council (TheoryDL project).

References

[1] Michael Fielding Barnsley and Lyman P Hurd. *Fractal image compression*, volume 1. AK peters Wellesley, 1993.

[2] PETER BLOEM. Fractal geometry. 2010.

[3] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

[4] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[5] Amit Daniely. Depth separation for neural networks. *arXiv preprint arXiv:1702.08489*, 2017.

[6] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

[7] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940, 2016.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Flip Korn, B-U Pagel, and Christos Faloutsos. On the" dimensionality curse" and the" self-similarity blessing". *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96–111, 2001.

[10] Hrushikesh N Mhaskar and Tomaso Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

[11] Guido Montúfar. Notes on the number of linear regions of deep neural networks. 2017.

[12] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[13] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[14] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[15] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[16] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

[17] Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. *arXiv preprint arXiv:1610.09887*, 2016.

[18] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. *arXiv preprint arXiv:1711.02114*, 2017.

[19] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[20] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. *arXiv preprint arXiv:1703.07950*, 2017.

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

[22] Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

A Proof of Theorem 1

To prove the theorem, we begin with two technical lemmas:

Lemma 1 *For every $\epsilon > 0$, there exists a neural-network of width $3d$ with two hidden-layers ($k = 3d, t = 3$) such that $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x}) = \boldsymbol{x}$ for $\boldsymbol{x} \in [0,1]^d$, and $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \mathbb{R}^d$ with $d(\boldsymbol{x}, [0,1]^d) = \min_{\boldsymbol{y} \in [0,1]^d} \|\boldsymbol{x} - \boldsymbol{y}\| > \epsilon$.*

Proof Let $N > 0$ be some constant, and observe the function:

$$f_i(\boldsymbol{x}) = \sigma(\sigma(x_i) - N\sum_{j=1}^{d}\sigma(-x_j) - N\sum_{j=1}^{d}\sigma(x_j - 1))$$

Notice that $f_i(\boldsymbol{x}) = x_i$ for $\boldsymbol{x} \in [0,1]^d$, and that $f_i(\boldsymbol{x}) = 0$ if $d(\boldsymbol{x}, [0,1]^d) > \epsilon$, when taking N to be large enough. Since $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \dots, f_d(\boldsymbol{x}))$ is a two hidden layer neural-network of width $3d$, the required follows. ■

Lemma 2 *For every $\gamma > 0$, there exists a neural-network of width $2d$ with two hidden-layers ($k = 2d, t = 3$) such that $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \notin [0,1]^d$, and $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in [\gamma, 1-\gamma]^d$.*

Proof Let $N > 0$ be some constant, and observe the function:

$$\tilde{f}(\boldsymbol{x}) = 1 - \sigma(1 - N\sum_{j=1}^{d}\sigma(\gamma - x_j) - N\sum_{j=1}^{d}\sigma(x_j - 1 + \gamma))$$

Notice that $\tilde{f}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in [\gamma, 1-\gamma]^d$, and that $\tilde{f}(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \notin [0,1]^d$, when taking N to be large enough. Since $\tilde{f}$ a two hidden layer neural-network of width $2d$, the required follows. ■

The next lemmas will show how a single block of the network operates on the set K_n:

Lemma 3 *There exists a neural-network of width $\max\{dr, 3d\}$ with two hidden-layers ($k = 3dr, t = 3$) such that for any n we have:*

1. $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(K_n) \subseteq K_{n-1}$
2. $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(K_1 \setminus K_n) \subseteq \mathcal{X} \setminus K_{n-1}$

Proof As an immediate corollary from Lemma 1, there exists $f : \mathbb{R}^d \to \mathbb{R}^d$, that can be implemented by a neural network with two hidden-layers and width $3d$, such that $f(\boldsymbol{x}) = \boldsymbol{x}$ for $\boldsymbol{x} \in K_0$ and $f(\boldsymbol{x}) = 0$ if $d(\boldsymbol{x}, K_0) > \frac{\epsilon}{2}$. Define the following function:

$$g(\boldsymbol{x}) = \sum_{i=1}^{r} f\left((\boldsymbol{M}^{(i)})^{-1}\boldsymbol{x} - (\boldsymbol{M}^{(i)})^{-1}\boldsymbol{v}^{(i)}\right) = \sum_{i=1}^{r} f\left(F_i^{-1}(\boldsymbol{x})\right)$$

Notice that for every $\boldsymbol{x} \in \mathcal{X}$ there is at most one $i \in [r]$ such that $f(F_i^{-1}(\boldsymbol{x})) > 0$. Indeed, assume there are $i \neq j \in [r]$ such that $f(F_i^{-1}(\boldsymbol{x})) > 0$ and $f(F_j^{-1}(\boldsymbol{x})) > 0$. Therefore, $d(F_i^{-1}(\boldsymbol{x}), K_0) \leq \frac{\epsilon}{2}$ and $d(F_j^{-1}(\boldsymbol{x}), K_0) \leq \frac{\epsilon}{2}$. Therefore, there exist $\boldsymbol{y}, \boldsymbol{z} \in K_0$ such that $\left\|F_i^{-1}(\boldsymbol{x}) - \boldsymbol{y}\right\| \leq \frac{\epsilon}{2}$ and $\left\|F_j^{-1}(\boldsymbol{x}) - \boldsymbol{z}\right\| \leq \frac{\epsilon}{2}$. From this we get:

$$\begin{aligned}
\|\boldsymbol{x} - F_i(\boldsymbol{y})\| &= \left\|\boldsymbol{x} - \boldsymbol{M}^{(i)}\boldsymbol{y} - \boldsymbol{v}^{(i)}\right\| \\
&= \left\|\boldsymbol{M}^{(i)}\left((\boldsymbol{M}^{(i)})^{-1}\boldsymbol{x} - (\boldsymbol{M}^{(i)})^{(-1)}\boldsymbol{v}^{(i)} - \boldsymbol{y}\right)\right\| \\
&\leq \left\|(\boldsymbol{M}^{(i)})^{-1}\boldsymbol{x} - (\boldsymbol{M}^{(i)})^{(-1)}\boldsymbol{v}^{(i)} - \boldsymbol{y}\right\| \\
&= \left\|F_i^{-1}(\boldsymbol{x}) - \boldsymbol{y}\right\| \leq \frac{\epsilon}{2}
\end{aligned}$$

where we use the fact that $\boldsymbol{M}^{(i)}$ is a contraction. Similarly, we get that $\|\boldsymbol{x} - F_j(\boldsymbol{z})\| \leq \frac{\epsilon}{2}$, so this gives us $\|F_i(\boldsymbol{y}) - F_j(\boldsymbol{z})\| \leq \epsilon$. Since $\boldsymbol{y}, \boldsymbol{z} \in K_0$, this is contradiction to Assumption 1.

We now show the following:

1. $g(K_n) \subseteq K_{n-1}$:

   Let $\boldsymbol{x} \in K_n$, and denote $i \in [r]$ the unique i for which $\boldsymbol{x} \in F_i(K_{n-1}) \subseteq F_i(K_0)$. From the properties of f, we get that $f(F_i^{-1}(\boldsymbol{x})) = F_i^{-1}(\boldsymbol{x})$ and $f(F_j^{-1}(\boldsymbol{x})) = 0$ for $j \neq i$, so $g(\boldsymbol{x}) = f(F_i^{-1}(\boldsymbol{x})) = F_i^{-1}(\boldsymbol{x}) \in K_{n-1}$.

2. $g(K_1 \setminus K_n) \subseteq \mathcal{X} \setminus K_{n-1}$:

   Let $\boldsymbol{x} \in K_1 \setminus K_n$ and assume by contradiction that $g(\boldsymbol{x}) \in K_{n-1}$. Let $i \in [r]$ be the unique i such that $\boldsymbol{x} \in F_i(K_0)$ and we have seen that in this case $g(\boldsymbol{x}) = F_i^{-1}(\boldsymbol{x})$, so $F_i^{-1}(\boldsymbol{x}) \in K_{n-1}$ and therefore $\boldsymbol{x} \in F_i(K_{n-1}) \subseteq K_n$ in contradiction to the assumption.

Since g can be implemented with a neural network of width $3dr$ and two hidden-layer, this completes the proof of the lemma. ■

Lemma 4 *There exists a neural-network of width $2dr$ with two hidden-layers ($k = 2dr, t = 3$) such that for any n we have:*

1. $\mathcal{N}_{\mathsf{W},\boldsymbol{B}}(\mathcal{X} \setminus K_1) = \{1\}$
2. $\mathcal{N}_{\mathsf{W},\boldsymbol{B}}(K_1^{\gamma}) = \{0\}$

Proof As a corollary of Lemma 2, there exists $\tilde{f} : \mathbb{R}^d \to \mathbb{R}$, a two hidden-layer neural-network of width $2d$, such that $\tilde{f}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \notin K_0$ and $\tilde{f}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in K_0^{\gamma}$. Now, define:

$$\tilde{g}(\boldsymbol{x}) = 1 - r + \sum_{i=1}^{r} \tilde{f}\left(F_i^{-1}(\boldsymbol{x})\right)$$

We show the following:

1. $\tilde{g}(\mathcal{X} \setminus K_1) = \{1\}$:

   Let $\boldsymbol{x} \notin K_1 = \cup_i F_i(K_0)$, then for every i we have $\boldsymbol{x} \notin F_i(K_0)$ and hence $F_i^{-1}(\boldsymbol{x}) \notin K_0$ so $\tilde{f}(F_i^{-1}(\boldsymbol{x})) = 1$ and so $\tilde{g}(\boldsymbol{x}) = 1$.

2. $\tilde{g}(K_1^\gamma) = \{0\}$:

   Let $\boldsymbol{x} \in K_1^\gamma$, and let i be the unique index such that $\boldsymbol{x} \in F_i(K_0)$. So we have $\tilde{f}(F_i^{-1}(\boldsymbol{x})) = 0$ and for all $j \neq i$ we have $\tilde{f}(F_j^{-1}(\boldsymbol{x})) = 1$, and therefore $\tilde{g}(\boldsymbol{x}) = 0$.

And $\tilde{g}$ can be implemented by a width $2dr$ two hidden-layer network. ■

Proof of Theorem 1 Let $g, \tilde{g}$ as defined in the previous lemmas. Denote $h_0 : \mathbb{R}^d \to \mathbb{R}^{d+1}$ the function:

$$h_0(\boldsymbol{x}) = [g(\boldsymbol{x}), \tilde{g}(\boldsymbol{x})]$$

and denote $h : \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ the function:

$$h(\boldsymbol{x}) = [g(\boldsymbol{x}_{1\ldots d}), x_{d+1} + \tilde{g}(\boldsymbol{x}_{1\ldots d})]$$

Denote h^n the composition of h on itself n times, and observe the network defined by $H = h^{n-1} \circ h_0$. Note that H satisfies the following properties:

1. For $\boldsymbol{x} \in K_n^\gamma$ we have $H(\boldsymbol{x})_{d+1} = 0$: indeed, by iteratively applying the previous lemmas, we get that $g^j(\boldsymbol{x}) \in K_1^\gamma$ for every $j \leq n-1$, and therefore $\tilde{g}(g^j(\boldsymbol{x})) = 0$ for every $j \leq n-1$. Observe that: $H(\boldsymbol{x})_{d+1} = \sum_{j=1}^{n-1} \tilde{g}(g^j(\boldsymbol{x})) = 0$.

2. For $\boldsymbol{x} \notin K_n$ we have $H(\boldsymbol{x})_{d+1} \geq 1$: there exists K_j such that $\boldsymbol{x} \in K_j \setminus K_{j+1}$, so by applying 3 we get $g^j(\boldsymbol{x}) \notin K_1$, so $\tilde{g}(g^j(\boldsymbol{x})) = 1$, and therefore $H(\boldsymbol{x})_{d+1} \geq 1$ (since the summation is over positive values).

Therefore, composing $H(\boldsymbol{x})$ with a linear threshold on $H(\boldsymbol{x})_{d+1}$ gives a network as required. Since ever block of H is has two hidden-layers of width $5dr$, we get that this network has depth $2n+1$ and width $5dr$. ■

B Proof of Corollary 2

Proof As mentioned, we can rewrite the IFS with r^s transformations, generating $K_{s\cdot\lfloor n/s\rfloor}$ in $\lfloor n/s \rfloor$ iterations. Therefore, using the construction of Theorem 1, we have a network of depth $2\lfloor \frac{n}{s} \rfloor$ and width $5dr^s$ that maps $K_{\lfloor n/s\rfloor}$ to K_0, and therefore maps K_n to $K_{n-\lfloor n/s\rfloor}$. Now, a two hidden-layer network of width at most dr^s can separate $K_{n-\lfloor n/s\rfloor}^\gamma$ from $\mathcal{X} \setminus K_{n-\lfloor n/s\rfloor}$. This constructs a network of depth $2\lfloor n/s \rfloor + 2$ and width $5dr^s$ that achieves the required. ■

C Proof of Theorem 4

Proof Using again [11], we get that the number of linear regions in $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ is r^{st}. This means that $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ crosses zero at most r^{st} times. Now, fix $n > j > st$, and notice that K_j is a union of r^j intervals, so $K_j = \cup_{i=1}^{r^j} I_i$, for intervals I_i. By our assumption, we get that for every i, $\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in I_i \textit{ and } y = -1\right] = p$ for some p, and from this we get:

$$\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in I_i \textit{ and } y = -1\right] = r^{-j}\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in K_j \textit{ and } y = -1\right]$$

We get that there are at most r^{st} intervals of K_j in which $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ crosses zero. Denote $J \subseteq [r^j]$ the subset of intervals on which $\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}})$ is constant, and for every $i \in J$ we denote $\hat{y}_i$ such that $\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(I_i)) = \{\hat{y}_i\}$. Notice that:

$$\begin{aligned}
&\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in K_j \textit{ and } y = 1\right] = \frac{1}{2} \\
&\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in K_j \textit{ and } y = -1\right] = 1 - P(j)
\end{aligned}$$

So the optimal choice for every $\hat{y}_i$ is 1. Then we have:

$$\begin{aligned}
\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)) \neq y\right] &= \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)) \neq -1 \textit{ and } x \notin K_j\right] \\
&\quad + \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)) \neq y \textit{ and } x \in K_j\right] \\
&\geq \sum_{i\in[r^j]} \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\text{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)) \neq y \textit{ and } x \in I_i\right] \\
&\geq \sum_{i\in J} \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\hat{y}_i \neq y \textit{ and } x \in I_i\right] \\
&\geq \sum_{i\in J} \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = -1 \textit{ and } x \in I_i\right] \\
&= |J| r^{-j}\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in K_j \textit{ and } y = -1\right] \\
&\geq (1 - r^{st-j})(1 - P(j))
\end{aligned}$$

■

D Proof of Theorem 5

Observe that for every n', we can write $C_{n'}$ as union of $2^{n'}$ intervals, so $C_{n'} = \cup_j I_j$. We can observe the distribution limited to each of these intervals, and get the following:

Lemma 5 *Let $\mathcal{D}_n$ be some cantor distribution (as defined in the paper). Then:*

$$\begin{aligned}
&\left|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[y \middle| x \in I_j\right]\right| \leq 2\left(P(n') - \frac{1}{2}\right) \\
&\left|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[xy \middle| x \in I_j\right]\right| \leq 2\left(P(n') - \frac{1}{2}\right)
\end{aligned}$$

Proof Let I_j be some interval of $C_{n'}$, and let c_j be the central point of I_j. Notice that by definition of the distribution we have:

$$\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = 1 \; and \; x \in I_j\right] = 2^{-n'-1}$$
$$\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = -1 \; and \; x \in I_j\right] = 2^{-n'-1}(1 - \sum_{i=1}^{n'} p_i) = 2^{-n'}(1 - P(n'))$$

So we get that $\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x \in I_j\right] = 2^{-n'}(\frac{3}{2} - P(n'))$, and therefore:

$$\begin{aligned}\left|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[y \middle| x \in I_j\right]\right| &= \left|\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = 1 \middle| x \in I_j\right] - \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = -1 \middle| x \in I_j\right]\right| \\ &= \left|(P(n') - \frac{1}{2})(\frac{3}{2} - P(n'))^{-1}\right| \\ &\le 2\left(P(n') - \frac{1}{2}\right)\end{aligned}$$

Notice that from the structure of the set C_n, the average of all the points in $I_j \cap C_n$ is exactly the central point c_j (this is due to the symmetry of the cantor set around its central point). Similarly, we get that each level of the negative distribution, $E_i := C_{i-1} \setminus C_i$, its average is also c_j. So we get:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[x \middle| x \in I_j \; and \; y = -1\right] = \mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[x \middle| x \in I_j \; and \; y = 1\right] = c_j$$

Therefore, we get that:

$$\begin{aligned}\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[xy \middle| x \in I_j\right] &= c_j\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = 1 \middle| x \in I_j\right] \\ &\quad - c_j\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[y = -1 \middle| x \in I_j\right] \\ &= c_j(P(n') - \frac{1}{2})(\frac{3}{2} - P(n'))^{-1}\end{aligned}$$

So we have:

$$\left|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[xy \middle| x \in I_j\right]\right| \le 2\left(P(n') - \frac{1}{2}\right)$$

■

Using this result, we get the following lemma:

Lemma 6 *Let $g : \mathbb{R} \to \mathbb{R}^k$, $f : \mathbb{R}^k \to \mathbb{R}$ two functions, and let $\boldsymbol{W} \in \mathbb{R}^{k\times k}$, $\boldsymbol{c} \in \mathbb{R}^k$, such that for every j, g is affine on I_j and f is affine on $\boldsymbol{W}g(I_j)+\boldsymbol{c} \subseteq \mathbb{R}^k$. For every j, denote $\boldsymbol{u}_j, \boldsymbol{v}_j, \boldsymbol{a}_j, \boldsymbol{b}_j \in \mathbb{R}^k$ such that for every $\boldsymbol{x} \in I_j$:*

$$g(x) = x\boldsymbol{u}_j + \boldsymbol{a}_j, \; f(\boldsymbol{W}g(x) + \boldsymbol{c}) = \boldsymbol{v}_j^\top(\boldsymbol{W}g(x) + \boldsymbol{c}) + \boldsymbol{b}_j$$

Assume that $\|\boldsymbol{u}_j\|_\infty, \|\boldsymbol{v}_j\|_\infty, \|\boldsymbol{a}_j\|_\infty, \|\boldsymbol{b}_j\|_\infty \le 1$. *Denote* $h : \mathbb{R} \to \mathbb{R}$ *s.t* $h(x) = f(\boldsymbol{W} g(x) + \boldsymbol{c})$. *Then the following holds:*

$$\left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \frac{\partial}{\partial \boldsymbol{W}} h(x) \middle| x \in C_{n'} \right] \right\|_{\max} \le 4 \left(P(n') - \frac{1}{2} \right)$$
$$\left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \frac{\partial}{\partial \boldsymbol{c}} h(x) \middle| x \in C_{n'} \right] \right\|_{\infty} \le 2 \left(P(n') - \frac{1}{2} \right)$$

Where for matrix $\boldsymbol{A}$ *we denote* $\|\boldsymbol{A}\|_{\max} = \max_{i,j} |a_{i,j}|$.

Proof For every $x \in I_j$ it holds that:

$$\frac{\partial}{\partial \boldsymbol{W}} h(x) = \frac{\partial}{\partial \boldsymbol{W}} \left[\boldsymbol{v}_j^\top (\boldsymbol{W}(\boldsymbol{u}_j x + \boldsymbol{a}_j) + \boldsymbol{c}) + \boldsymbol{b}_j \right] = \boldsymbol{u}_j \boldsymbol{v}_j^\top x + \boldsymbol{a}_j \boldsymbol{v}_j^\top$$
$$\frac{\partial}{\partial \boldsymbol{c}} h(x) = \frac{\partial}{\partial \boldsymbol{c}} \left[\boldsymbol{v}_j^\top (\boldsymbol{W}(\boldsymbol{u}_j x + \boldsymbol{a}_j) + \boldsymbol{c}) + \boldsymbol{b}_j \right] = \boldsymbol{v}_j$$

Using Lemma 5, we get:

$$\begin{aligned}
\left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \frac{\partial}{\partial \boldsymbol{W}} h(x) \middle| x \in I_j \right] \right\|_{\max} &= \left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y(\boldsymbol{u}_j \boldsymbol{v}_j^\top x + \boldsymbol{a}_j \boldsymbol{v}_j^\top) \middle| x \in I_j \right] \right\|_{\max} \\
&\le \left| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-yx \middle| x \in I_j \right] \right| \cdot \left\| \boldsymbol{u}_j \boldsymbol{v}_j^\top \right\|_{\max} \\
&+ \left| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \middle| x \in I_j \right] \right| \cdot \left\| \boldsymbol{a}_j \boldsymbol{v}_j^\top \right\|_{\max} \\
&\le 4 \left(P(n') - \frac{1}{2} \right)
\end{aligned}$$

$$\begin{aligned}
\left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \frac{\partial}{\partial \boldsymbol{c}} h(x) \middle| x \in I_j \right] \right\|_{\infty} &= \left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \boldsymbol{v}_j \middle| x \in I_j \right] \right\|_{\infty} \\
&= \left| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \middle| x \in I_j \right] \right| \cdot \|\boldsymbol{v}_j\|_\infty \le 2 \left(P(n') - \frac{1}{2} \right)
\end{aligned}$$

Finally, from this we get:

$$\begin{aligned}
\left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \frac{\partial}{\partial \boldsymbol{W}} h(x) \middle| x \in C_{n'} \right] \right\|_{\max} &\le \sum_j \mathbb{P}\left[x \in I_j \right] \left\| \mathbb{E}_{(x,y)\sim\mathcal{D}_F} \left[-y \frac{\partial}{\partial \boldsymbol{W}} h(x) \middle| x \in I_j \right] \right\|_{\max} \\
&\le 4 \left(P(n') - \frac{1}{2} \right)
\end{aligned}$$

$$\left\|\mathbb{E}_{(x,y)\sim\mathcal{D}_F}\left[-y\frac{\partial}{\partial\boldsymbol{c}}h(x)\Big|x\in C_{n'}\right]\right\|_\infty \le \sum_j \mathbb{P}\left[x\in I_j\right]\left\|\mathbb{E}_{(x,y)\sim\mathcal{D}_F}\left[-y\frac{\partial}{\partial\boldsymbol{c}}h(x)\Big|x\in I_j\right]\right\|_\infty$$
$$\le 2\left(P(n')-\frac{1}{2}\right)$$

■

Now, we need to show that with high probability over the initialization of the network, every layer is affine on I_j-s.

Lemma 7 *Fix $\delta\in(0,1)$, and let $s\le k$. Let $g:\mathbb{R}\to\mathbb{R}^s$ such that for every j, g is affine and non-expansive on I_j (w.r.t to $\|\cdot\|_\infty$). Let $\boldsymbol{W}\in\mathbb{R}^{k\times s}$ a random matrix such that every entry is initialized uniformly from $[-\frac{1}{2s},\frac{1}{2s}]$, and let $b>2k^2\left(\frac{2}{3}\right)^{n'}\delta^{-1}$, some fixed bias. Denote $h(x):=\psi(\boldsymbol{W}g(x)+b)$, for some ψ that is affine on every interval that is bounded away from zero. Then with probability at least $1-\delta$, for every j, $h(x)$ is affine and non-expansive on I_j.*

Proof Denote $\boldsymbol{w}_i\in\mathbb{R}^k$ the i-th row of $\boldsymbol{W}$. Fix some j, and denote c_j the central point of I_j. We show that:
$$\mathbb{P}_{\boldsymbol{w}_i\sim\mathcal{U}([-\frac{1}{2s},\frac{1}{2s}]^s)}\left[|\boldsymbol{w}_i^\top g(c_j)+b|\le 3^{-n'}\right]\le\frac{\delta}{k2^{n'}}$$
If $\|g(c_j)\|_\infty\le b$ then $|\boldsymbol{w}_i^\top g(c_j)|\le\|\boldsymbol{w}_i\|_1\|g(c_j)\|_\infty\le\frac{b}{2}$ and therefore $|\boldsymbol{w}_i^\top g(c_j)+b|\ge\frac{b}{2}>3^{-n'}$. So we can assume $\|g(c_j)\|_\infty>b$, and let $\ell\in[k]$ be some index such that $g(c_j)_\ell>b$. Now, fix some values for $w_{i,1},\dots,w_{i,\ell-1},w_{i,\ell+1},\dots,w_{i,k}$, and observe the distribution of $\boldsymbol{w}_i^\top g(c_j)+b$ (with respect to the randomness of $w_{i,\ell}$). Since $w_{i,\ell}$ is uniformly distributed in $[-\frac{1}{2s},\frac{1}{2s}]$, we get that this is a uniform distribution over some interval J with $|J|\ge\frac{b}{s}$. From this we get:

$$\begin{aligned}\mathbb{P}_{w_{i,\ell}\sim\mathcal{U}([-\frac{1}{2s},\frac{1}{2s}])}\left[|\boldsymbol{w}_i^\top g(c_j)+b|\le 3^{-n'}\right]&=\mathbb{P}_{x\sim\mathcal{U}(J)}\left[x\in[-3^{-n'},3^{-n'}]\right]\\&=|J\cap[-3^{-n'},3^{-n'}]|/|J|\\&\le|[-3^{-n'},3^{-n'}]|/|J|\\&=\frac{2s3^{-n'}}{b}\le\frac{\delta}{k2^{n'}}\end{aligned}$$

Since there are $2^{n'}$ intervals I_j and k rows in $\boldsymbol{W}$, using the union bound we get that with probability at least $1-\delta$, we have for all $j\in[2^{n'}]$ that $\|\boldsymbol{W}g(c_j)+b\|_\infty>3^{-n'}$. Since we have $|I_j|=3^{-n'}$, and since g is non-expansive, this means that the set $\boldsymbol{W}g(I_j)+b$ does not cross zero at any of its coordinates. Indeed, assume there exists $i\in[k]$ such that $\boldsymbol{w}_i^\top g(I_j)+b$ crosses zero, and assume w.l.o.g that $\boldsymbol{w}_i^\top g(I_j)+b>0$. Then there exists $x\in I_j$ with $\boldsymbol{w}_i^\top g(x)+b\le 0$ and therefore:

$$3^{-n'}<|\boldsymbol{w}_i^\top g(x)-\boldsymbol{w}_i^\top g(c_j)|\le\|\boldsymbol{w}_i\|_1\|g(x)-g(c_j)\|_\infty\le|x-c_j|\le\frac{1}{2}3^{-n'}$$

and we reach contradiction.

Since ψ is affine on intervals that are bounded away from zero, we get that $h(x)=\psi(\boldsymbol{W}g(x)+b)$ is affine on all I_j.

To show that h is non-expansive on I_j, let $x, y \in I_j$, and from the fact that g is non-expansive we have $\|g(x) - g(y)\|_\infty \leq |x - y|$. Since we showed that ψ is affine on $\boldsymbol{W}g(I_j) + b$, we get:

$$|h(x) - h(y)| = |\psi(\boldsymbol{W}g(x) + b) - \psi(\boldsymbol{W}g(y) + b)| = |\psi(\boldsymbol{W}(g(x) - g(y)))| \leq |\boldsymbol{W}(g(x) - g(y))|$$

Therefore, for every i we get:

$$|h(x)_i - h(y)_i| = |\boldsymbol{w}_i^\top (g(x) - g(y))| \leq \|\boldsymbol{w}_i\|_1 \|g(x)_j - g(y)_j\|_\infty \leq |x - y|$$

which completes the proof. ■

Lemma 8 *Let $g : \mathbb{R} \to \mathbb{R}^s$ such that $\|g(x)\|_\infty \leq 1$ for every $x \in [0, 1]$. Let $\boldsymbol{W} \in \mathbb{R}^{k \times s}$ a random matrix such that every entry is initialized uniformly from $[-\frac{1}{2s}, \frac{1}{2s}]$, and let $0 < b \leq \frac{1}{2}$ some bias. Denote $h(x) := \sigma(\boldsymbol{W}g(x) + b)$. Then $\|h(x)\|_\infty \leq 1$ for every $x \in [0, 1]$.*

Proof As before, we denote $\boldsymbol{w}_i$ the i-th row of $\boldsymbol{W}$, then for every $x \in [0, 1]$ we get:

$$\|h(x)\|_\infty \leq \max_i |\boldsymbol{w}_i^\top g(x) + b| \leq \max_i \|\boldsymbol{w}_i\|_1 \|g(x)\|_\infty + b \leq 1$$

■

Iteratively applying this lemma gives a bound on the norm of any hidden representation in the network:

Lemma 9 *Assume we initialize a neural-network $\mathcal{N}_{\mathbf{W}, \boldsymbol{B}}$ as described in Theorem 5, and denote $\mathcal{N}_{\mathbf{W}, \boldsymbol{B}} = g^{(t)} \circ \cdots \circ g^{(1)}$. Denote $G^{(t')} = g^{(t')} \circ \cdots \circ g^{(1)}$, the output of the layer t'. Then for every layer t' and for every $x \in [0, 1]$ we get that: $\left\|G^{(t')}(x)\right\|_\infty \leq 1$.*

Proof First, $g^{(1)}(x) = \sigma(\boldsymbol{w}^{(1)}x + \boldsymbol{b}^{(1)})$, where $\boldsymbol{w}^{(1)} \sim \mathcal{U}([-\frac{1}{2}, \frac{1}{2}]^k)$ and $\boldsymbol{b}^{(1)} = [\frac{1}{2}, \dots, \frac{1}{2}]$, so for every $x \in [0, 1]$ we have:

$$\left\|g^{(1)}(x)\right\|_\infty \leq \left\|\boldsymbol{w}^{(1)}\right\|_\infty |x| + \frac{1}{2} \leq 1$$

Now, from Lemma 8, if $\left\|G^{(t')}(x)\right\|_\infty \leq 1$ for $x \in [0, 1]$, then $\left\|G^{(t'+1)}(x)\right\|_\infty \leq 1$ for $x \in [0, 1]$. By induction we get that $\left\|G^{(t')}(x)\right\|_\infty$ for $x \in [0, 1]$ for every $t' \leq t - 1$. Finally, we have $\boldsymbol{w}^{(t)} \sim \mathcal{U}([-\frac{1}{2k}, \frac{1}{2k}])$ and $b^{(t)} = \frac{1}{2}$, and this gives us for every $x \in [0, 1]$:

$$|\mathcal{N}_{\mathbf{W}, \boldsymbol{B}}(x)| \leq |\boldsymbol{w}^{(t)} G^{(t-1)}(x) + b^{(t)}| \leq \left\|\boldsymbol{w}^{(t)}\right\|_1 \left\|G^{(t-1)}(x)\right\|_\infty + b^{(t)} \leq 1$$

■

We also show that the gradients are bounded on all the examples in the distribution:

Lemma 10 *Assume we initialize a neural-network $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ as described in Theorem 5. Then for every layer t', and every example $x \in [0,1]$ we have:*

$$\left\| \frac{\partial}{\partial \boldsymbol{W}^{(t')}} \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) \right\|_{\max} \leq 1$$
$$\left\| \frac{\partial}{\partial \boldsymbol{b}^{(t')}} \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) \right\|_{\infty} \leq 1$$

Proof Recall that we denote for every $x \in [0,1]$ the output of the layer t' to be $x^{(t')} = G^{(t')}$. Denote $\boldsymbol{D}^{(t')} = \text{diag}(\sigma'(\boldsymbol{W}^{(t')} x^{(t')}))$. We calculate the gradient of the weights at layer t':

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{W}^{(t')}} \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) &= \frac{\partial}{\partial \boldsymbol{W}^{(t')}} g^{(t)} \circ \cdots \circ g^{(t')}(\sigma(\boldsymbol{W}^{(t')} x^{(t'-1)} + \boldsymbol{b}^{(t')})) \\ &= (\boldsymbol{w}^{(t)})^\top D^{(t-1)} \boldsymbol{W}^{(t-1)} \cdots D^{(t'+1)} \boldsymbol{W}^{(t'+1)} D^{(t')} x^{(t'-1)} \end{aligned}$$

Denote $\|\cdot\|_\infty^{OP}$ the operator norm induced by ℓ_∞, and we get (using the properties of the weights initialization):

$$\begin{aligned} \left\| D^{(t-1)} \boldsymbol{W}^{(t-1)} \cdots D^{(t'+1)} \boldsymbol{W}^{(t'+1)} D^{(t')} x^{(t'-1)} \right\|_\infty &\leq \left\| D^{(t-1)} \right\|_\infty^{OP} \left\| \boldsymbol{W}^{(t-1)} \right\|_\infty^{OP} \cdots \\ &\quad \cdot \left\| D^{(t'+1)} \right\|_\infty^{OP} \left\| \boldsymbol{W}^{(t'+1)} \right\|_\infty^{OP} \left\| D^{(t')} \right\|_\infty^{OP} \left\| x^{(t'-1)} \right\|_\infty \\ &\leq 1 \end{aligned}$$

And therefore: $\left\| \frac{\partial}{\partial \boldsymbol{W}^{(t')}} \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) \right\|_{\max} \leq 1$ Finally, we calculate the gradient of the bias at layer t':

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{b}^{(t')}} \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) &= \frac{\partial}{\partial \boldsymbol{b}^{(t')}} g^{(t)} \circ \cdots \circ g^{(t')}(\sigma(\boldsymbol{W}^{(t')} x^{(t'-1)} + \boldsymbol{b}^{(t')})) \\ &= (\boldsymbol{w}^{(t)})^\top D^{(t-1)} \boldsymbol{W}^{(t-1)} \cdots D^{(t'+1)} \boldsymbol{W}^{(t'+1)} D^{(t')} \end{aligned}$$

And since $\left\| \boldsymbol{w}^{(t)} \right\|_\infty \leq 1$ we get similarly to above that $\left\| \frac{\partial}{\partial \boldsymbol{b}^{(t')}} \mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) \right\|_\infty \leq 1$. ■

Proof of Theorem 5. Denote each layer of the network by $g^{(i)}$, so we have: $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x) = g^{(t)} \circ \cdots \circ g^{(1)}(x)$. We show that two things hold on initialization:

1. $|\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)| \leq 1$ for $x \in [0,1]$: immediately from Lemma 9.

2. With probability at least $1 - \delta$, for every j, $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}$ is affine on I_j:

   Denote $\hat{\delta} = \frac{\delta}{t}$, and notice that by the choice of n', we get that $2k^2 \left(\frac{2}{3}\right)^{n'} \hat{\delta}^{-1} < \frac{1}{2} = b$. Therefore, since σ is affine on all intervals away from zero, we can apply Lemma 7 on all the hidden layers of the network (choosing $s = 1, g = id$ for the first layer and $s = k, g = g^{(t')} \circ g^{(1)}$ for the rest), and use union bound to get the required.

Now, to prove the theorem, observe that since $\mathcal{D}_n$ is supported on $[0,1]\times\{\pm1\}$, we get that upon initialization with probability 1 for $(x,y)\sim\mathcal{D}_n$ we have: $\max\{1-y\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x),0\} = 1-y\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)$. Since $\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)$ is affine on every I_j w.p $1-\delta$, using Lemma 6 we get that in such case:

$$\begin{aligned}
\left\|\frac{\partial}{\partial\mathbf{W}}\mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}})\right\|_{\max} &= \left\|\frac{\partial}{\partial\mathbf{W}}\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[\max\{1-y\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x),0\}\right]\right\|_{\max}\\
&= \left\|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[-y\frac{\partial}{\partial\mathbf{W}}\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)\right]\right\|_{\max}\\
&\le \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x\in C_{n'}\right]\cdot\left\|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[-y\frac{\partial}{\partial\mathbf{W}}\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)\middle|x\in C_{n'}\right]\right\|_{\max}\\
&+ \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x\notin C_{n'}\right]\cdot\left\|\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[-y\frac{\partial}{\partial\mathbf{W}}\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x)\middle|x\notin C_{n'}\right]\right\|_{\max}\\
&\le 4\left(P(n')-\frac{1}{2}\right)+\left(P(n')-\frac{1}{2}\right) = 5\left(P(n')-\frac{1}{2}\right)
\end{aligned}$$

and similarly we get $\left\|\frac{\partial}{\partial\boldsymbol{B}}\mathcal{L}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}})\right\|_\infty \le 3\left(P(n')-\frac{1}{2}\right)$.

To show that $\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x))\neq y\right] \ge \left(\frac{3}{2}-P(n')\right)(1-P(n'))$, observe that the sign function is affine on intervals bounded away from zero. We can use Lemma 7 on the final layer, which shows that $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}})$ is affine on the intervals I_j, so for every I_j we get either $\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(I_j)) = \{\hat{y}_j\}$ for some $\hat{y}_j\in\{\pm1\}$. Now, using Lemma 6 we get that:

$$\begin{aligned}
\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x))\neq y\middle|x\in I_j\right] &= \mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[\frac{1}{2}-\frac{1}{2}y\hat{y}_j\middle|x\in I_j\right]\\
&= \frac{1}{2}-\frac{1}{2}\hat{y}_j\mathbb{E}_{(x,y)\sim\mathcal{D}_n}\left[y\middle|x\in I_j\right]\\
&\ge 1-P(n')
\end{aligned}$$

And from this we get:

$$\begin{aligned}
\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x))\neq y\right] &= \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x\in C_{n'}\right]\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x))\neq y\middle|x\in C_{n'}\right]\\
&+ \mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[x\notin C_{n'}\right]\mathbb{P}_{(x,y)\sim\mathcal{D}_n}\left[\operatorname{sign}(\mathcal{N}_{\mathbf{W},\boldsymbol{B}}(x))\neq y\middle|x\notin C_{n'}\right]\\
&\ge \left(\frac{3}{2}-P(n')\right)(1-P(n'))
\end{aligned}$$

■

E Proof of Corollary 4

Proof Denote $a = 2\log^{-1}(\frac{3}{2}), b = 2\log^{-1}(\frac{3}{2})\log(\frac{4k^2}{\delta})+1$, and from Lemma A.2 in [19], we get that if $t > 4a\log(2a)+2b$ then $t > a\log(t)+b$. Choosing $n=\frac{t}{2}$ gives $n > \log^{-1}(\frac{3}{2})\log(\frac{4tk^2}{\delta})+1$, so applying Theorem 5 shows that $\mathcal{D}_F^n$ satisfies 3. Theorem 1 immediately gives 1. Note that $\mathcal{D}_F^n$ can be realized only by functions with at least $2^{n-1}+1$ linear regions. Shallow networks on $\mathbb{R}$ of width k have at most $k+1$ linear regions, so this gives 2. ■

F Experimental Results

The following tables summarize the results of all the experiments that are detailed in Section 6.

Table 1: Performance of different network architectures on various fractal distributions of depth 5, with different fractal structures.

Depth / Width	10	20	50	100	200	400
Sierpinsky Triangle						
1	0.78	0.82	0.88	0.87	0.89	0.90
2	0.86	0.91	0.93	0.94	0.95	0.95
3	0.89	0.92	0.96	0.96	0.97	0.97
4	0.87	0.94	0.97	0.97	0.97	0.98
5	0.89	0.94	0.96	0.97	0.97	0.98
2D Cantor Set						
1	0.61	0.69	0.72	0.73	0.72	0.74
2	0.72	0.81	0.82	0.86	0.86	0.87
3	0.78	0.84	0.88	0.92	0.93	0.93
4	0.82	0.86	0.91	0.95	0.97	0.97
5	0.81	0.87	0.95	0.97	0.99	0.98
Pentaflake						
1	0.66	0.65	0.67	0.70	0.76	0.76
2	0.71	0.73	0.79	0.81	0.82	0.83
3	0.73	0.78	0.83	0.84	0.85	0.86
4	0.76	0.79	0.85	0.87	0.88	0.88
5	0.76	0.81	0.86	0.88	0.87	0.90
Vicsek						
1	0.59	0.60	0.63	0.66	0.67	0.68
2	0.64	0.70	0.72	0.75	0.76	0.75
3	0.69	0.72	0.77	0.79	0.81	0.82
4	0.71	0.74	0.79	0.82	0.83	0.84
5	0.70	0.77	0.82	0.84	0.86	0.86

Table 2: Performance of depth 5 network on the different fractal structure (of depth 5), with varying approximation curves.

Curve # / Width	10	20	50	100	200	400
Sierpinsky Triangle						
1	0.89	0.94	0.96	0.97	0.97	0.98
2	0.89	0.94	0.96	0.97	0.97	0.97
3	0.78	0.94	0.96	0.97	0.97	0.97
4	0.79	0.92	0.96	0.96	0.97	0.97
5	0.76	0.89	0.96	0.97	0.97	0.97
6	0.76	0.90	0.97	0.97	0.98	0.98
2D Cantor Set						
1	0.81	0.87	0.95	0.97	0.99	0.98
2	0.70	0.85	0.92	0.94	0.94	0.97
3	0.62	0.73	0.75	0.80	0.91	0.89
4	0.53	0.65	0.77	0.77	0.84	0.93
5	0.57	0.61	0.65	0.69	0.76	0.73
6	0.53	0.64	0.66	0.78	0.71	0.61
Pentaflake						
1	0.76	0.81	0.86	0.88	0.87	0.90
2	0.59	0.68	0.77	0.78	0.80	0.84
3	0.54	0.57	0.64	0.63	0.72	0.64
4	0.53	0.55	0.58	0.61	0.65	0.68
5	0.52	0.52	0.52	0.55	0.60	0.57
6	0.52	0.52	0.52	0.53	0.56	0.54
Vicsek						
1	0.70	0.77	0.82	0.84	0.86	0.86
2	0.59	0.61	0.67	0.69	0.71	0.71
3	0.56	0.55	0.58	0.64	0.64	0.65
4	0.51	0.52	0.54	0.56	0.58	0.59
5	0.52	0.52	0.52	0.55	0.53	0.57
6	0.53	0.51	0.51	0.55	0.58	0.59