Is Deeper Better only when Shallow is Good?

Eran Malach and Shai Shalev-Shwartz

Mobileye and The Hebrew University of Jerusalem

Simons Institute, Berkeley, 2019
https://joelgrus.com/2016/05/23/fizz-buzz-in-tensorflow/

interviewer: OK, so I need you to print the numbers from 1 to 100, except that if the number is divisible by 3 print "fizz", if it’s divisible by 5 print ”buzz”, and if it’s divisible by 15 print ”fizzbuzz”.
Do you need help getting started?

me: No, no, I’m good. So let’s start with some standard imports:

```python
import numpy as np
import tensorflow as tf
```
interviewer: OK, so I need you to print the numbers from 1 to 100, except that if the number is divisible by 3 print "fizz", if it’s divisible by 5 print "buzz", and if it’s divisible by 15 print "fizzbuzz". Do you need help getting started?

me: No, no, I’m good. So let’s start with some standard imports:

```python
import numpy as np
import tensorflow as tf
```

Postscript: I didn’t get the job. So I tried actually running this, and it turned out it got some of the outputs wrong! Thanks a lot, machine learning!
interviewer: OK, so I need you to print the numbers from 1 to 100, except that if the number is divisible by 3 print "fizz", if it’s divisible by 5 print "buzz", and if it’s divisible by 15 print "fizzbuzz". Do you need help getting started?

me: No, no, I’m good. So let’s start with some standard imports:

```python
import numpy as np
import tensorflow as tf
```

Postscript: I didn’t get the job. So I tried actually running this, and it turned out it got some of the outputs wrong! Thanks a lot, machine learning!
I guess maybe I should have used a deeper network …
Basic question: on which distributions deeper networks are much better than shallow ones?
Basic question: on which distributions deeper networks are much better than shallow ones?

Several recent results show

Depth Separation

There exist functions which can be expressed by a small deep network but must have an exponential width in order to be expressed by a shallow network

Outline

Main Claim

Strong depth separation $\Rightarrow$ Gradient based Algorithms fail

1. Case study: Fractal Distributions

2. Depth Separation

3. Approximation Curve and Strong Depth Separation

4. Success of SGD depends on the Approximation Curve
Fractals

- **Iterated Function System:**
  
  \[ K_0 = [-1, 1]^d \]
  
  \[ K_n = F_1(K_{n-1}) \cup \ldots \cup F_r(K_{n-1}) \]

- We assume \( F_i \) are affine, invertible, contractive, and for \( i \neq j \), the images of \( F_i \) and \( F_j \) are disjoint.

- The “depth” of the fractal is \( n \)

- Example: \( F_i(x) = c_i + \frac{1}{4}(x - c_i) \) for \( c_i \in \{\pm 1\}^2 \)
A “fractal distribution” is a distribution in which positive examples are sampled from the set $K_n$ and negative examples are sampled from its complement.

Examples:

- Cantor
- Sierpinsky
- Vicsek
- Pentaflake
**Theorem**

Consider an IFS over \([-1, 1]^d\) with \(r\) generating functions and depth \(n\). For any fractal distribution \(D_n\) there exists a ReLU feed forward network of depth \(2n + 1\) and width \(5dr\) which realizes \(D_n\).
**Theorem**

Consider an IFS over \([-1, 1]^d\) with \(r\) generating functions and depth \(n\). For any fractal distribution \(D_n\) there exists a ReLU feed forward network of depth \(2n + 1\) and width \(5dr\) which realizes \(D_n\).

**Proof by induction:**

- **Basis:** A shallow ReLU network can approximate \(I_0(x) = 1_{x \in K_0}\).
- **Suppose** we have a deep network expressing: \(I_{n-1}(x) = 1_{x \in K_{n-1}}\).
- **Recall:** \(K_n = F_1(K_{n-1}) \cup \ldots \cup F_r(K_{n-1})\) and \(F_i\) are affine, invertible, and have disjoint images.
- **Take** \(x \in K_n\), then there’s \(z \in K_{n-1}\) and \(i\) s.t. \(x = F_i(z)\), or equivalently, \(z = F_i^{-1}(x)\).
- **Therefore,** \(\left[\sum_i I_{n-1}(F_i^{-1}(x))\right]_+ - \left[\sum_i I_{n-1}(F_i^{-1}(x)) - 1\right]_+ = 1_{x \in K_n}\).
Theorem

If $D_n$ has non-zero probability in any area of $K_n$, then a network of depth $t$ must have a width of at least $\frac{d}{e} r^\frac{n}{td}$ to realize $D_n$. 

Proof idea:
A network of width $k$ and depth $t$ has at most $(ek/d)^{td}$ linear regions. To realize the fractal distribution, we need $r^n$ linear regions.
Depth Separation

**Theorem**

If $D_n$ has non-zero probability in any area of $K_n$, then a network of depth $t$ must have a width of at least $\frac{d}{e} r^{\frac{n}{td}}$ to realize $D_n$.

**Proof idea:**

- A network of width $k$ and depth $t$ has at most $(ek/d)^{td}$ linear regions.
- To realize the fractal distribution, we need $r^n$ linear regions.
Main Claim

Strong depth separation $\Rightarrow$ Gradient based Algorithms fail

1. Case study: Fractal Distributions

2. Depth Separation

3. Approximation Curve and Strong Depth Separation

4. Success of SGD depends on the Approximation Curve
We saw: a network of depth $O(n)$ can express a depth $n$ fractal, but a shallower network requires exponential width to fully realize the distribution.

**Approximation curve:** How much of the negative examples are on the fine details of the fractal:

$$P(j) := 1 - L_{D_n}(1_{x \in K_j}) := 1 - \mathbb{P}_{(x,y) \sim D_n} [x \in K_j \land y = -1]$$

Note: $P(0) = 1/2$, $P(n) = 1$, and $P$ is monotonically increasing.
Approximation Curve: coarse vs. fine

\[ P(j) = 1 - L_{D_n}(1_{x \in K_j}) \]
The following theorem shows that with reasonable width, the error of a depth $\Theta(j)$ network is roughly $1 - P(j)$

**Theorem**

*Fix a depth $n$ distribution with approximation curve $P$. Then, for every $j$*

1. **For a depth $t = 2j + 2$ and width $k = 5dr$ network we have**

   $$L_{D_n}(H_{t,k}) \leq (1 - P(j))$$

2. **For every $s$, if $k < r^s$ and $t < j/s$ then**

   $$(1 - r^{st-j})(1 - P(j)) \leq L_{D_n}(H_{t,k})$$
Main Claim

Strong depth separation $\Rightarrow$ Gradient based Algorithms fail

1. Case study: Fractal Distributions

2. Depth Separation

3. Approximation Curve and Strong Depth Separation

4. Success of SGD depends on the Approximation Curve
One dimensional Cantor Fractal with “Fine” Distribution

- \(C_0 = [0, 1]\) and \(C_n = F_1(C_{n-1}) \cup F_2(C_{n-1})\), where \(F_1(x) = \frac{1}{3} - \frac{1}{3}x\) and \(F_2(x) = \frac{2}{3} + \frac{1}{3}x\)
- “Fine” cantor distributions of growing depth. Negative areas in orange, positive in blue.
Theorem

Consider a depth \( t \), width \( k \), network, and suppose the weights, \( W \), are initialized randomly in the “normal” way. Consider a depth \( n \), one-dimensional Cantor fractal, and let \( j = \lceil \log(tk^2/\delta) \rceil \). Then, with probability \( > 1 - \delta \), all elements of the gradient at \( W \) are of magnitude

\[
< 5(P(j) - \frac{1}{2}).
\]

Corollary: gradient descent is likely to fail on every cantor distribution with strong depth separation, even though the deep network is expressive enough.
Success of SGD depends on the Approximation Curve

Learning depth 5 network on 2D cantor set of depth 5, with different approximation curves.

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 $P(n)$) 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.

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, distributions with "fine" approximation curves cause a drastic degradation in performance.

Figure 6: The effect of depth on learning CIFAR-10, line colors as in figure 4. We train CNNs with Adam for 60K steps. All layers are 5x5 Convolutions with ReLU activation, except the readout layer. We perform max-pool only in the first two layers. We use augmentations and training pipeline in [20].

We perform the same experiments with different fractal structures (figure 1 shows these distributions). Tables 1, 2 in the appendix summarize the results. We note that the effect of depth can be seen clearly in all fractal structures. The effect of the approximation curve is observed in all fractals, except the Sierpinsky Triangle (generated with 3 transformations), where 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 small number of linear regions, making the problem overall relatively easy.

Finally, we want to show that the results given in this paper are interesting beyond the scope of our admittedly synthetic fractal distributions. We note that the use of fractal distributions is favorable from a theoretical perspective, as it allows us to develop crisp analysis and insightful results. On the other hand, it may raise a valid concern regarding the applicability of these results to real-world scenarios. To address this concern, we performed similar experiments on the CIFAR-10 data, studying the effect of width and depth on the performance of neural-networks on real data. The results are shown in figure 6. Notice that the trends on the CIFAR data resemble the behavior on the "coarse" fractal distributions. Importantly, note that the CIFAR data does not exhibit a strong depth separation, as depth gives only gradual improvement in performance. That is, while deeper networks indeed exhibit better performance, a shallow network already gives a good approximation. A similar behavior is observed even on the ImageNet dataset (see fig. 2 in [1]).
The effect of depth on learning CIFAR-10.

We train CNNs with Adam for 60K steps. All layers are 5x5 Convolutions with ReLU activation, except the readout layer.

Line colors correspond to different network depth.
Fractal distributions are natural for studying depth efficiency of deep learning.

The “approximation curve” is correlated with how much going deeper really helps.

Strong depth separation: shallow networks perform like random guess while deeper networks realize the distribution.

Conjecture: gradient based algorithms fail when there is strong depth separation. In other words, deep is better only when shallow is also good.
Conjecture:

- Let $\mathcal{H}$ be all functions which cannot be approximated by a shallow network. Then:
  1. For each $f \in \mathcal{H}$ there exists a distribution $D_f$ on $\mathcal{X} \times \{\pm 1\}$ for which $f$ achieves zero loss while the best shallow network achieves a loss $> 1/2 - \epsilon$.
  2. For every such $D_f$, gradient-descent fails to learn a deep network.