Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks

Itay Safran [1] **Ohad Shamir** [1]

Abstract

We provide several new depth-based separation results for feed-forward neural networks, proving that various types of simple and natural functions can be better approximated using deeper networks than shallower ones, even if the shallower networks are much larger. This includes indicators of balls and ellipses; non-linear functions which are radial with respect to the L_1 norm; and smooth non-linear functions. We also show that these gaps can be observed experimentally: Increasing the depth indeed allows better learning than increasing width, when training neural networks to learn an indicator of a unit ball.

1. Introduction

Deep learning, in the form of artificial neural networks, has seen a dramatic resurgence in the past recent years, achieving great performance improvements in various fields of artificial intelligence such as computer vision and speech recognition. While empirically successful, our theoretical understanding of deep learning is still limited at best.

An emerging line of recent works has studied the *expressive power* of neural networks: What functions can and cannot be represented by networks of a given architecture (see related work section below). A particular focus has been the trade-off between the network's width and depth: On the one hand, it is well-known that large enough networks of depth 2 can already approximate any continuous target function on $[0,1]^d$ to arbitrary accuracy (Cybenko, 1989; Hornik, 1991). On the other hand, it has long been evident that deeper networks tend to perform better than shallow ones, a phenomenon supported by the intuition that depth, providing compositional expressibility, is necessary for efficiently representing some functions. Moreover, recent empirical evidence suggests that standard feedforward deep networks are harder to optimize than shallower networks which lead to worse training error and testing error (He et al., 2015).

To demonstrate the power of depth in neural networks, a clean and precise approach is to prove the existence of functions which can be expressed (or well-approximated) by moderately-sized networks of a given depth, yet cannot be approximated well by shallower networks, even if their size is much larger. However, the mere existence of such functions is not enough: Ideally, we would like to show such depth separation results using *natural*, interpretable functions, of the type we may expect neural networks to successfully train on. Proving that depth is necessary for such functions can give us a clearer and more useful insight into what various neural network architectures can and cannot express in practice.

In this paper, we provide several contributions to this emerging line of work. We focus on standard, vanilla feedforward networks (using some fixed activation function, such as the popular ReLU), and measure expressiveness directly in terms of approximation error, defined as the expected squared loss with respect to some distribution over the input domain. In this setting, we show the following:

- We prove that the indicator of the Euclidean unit ball, $\mathbf{x} \mapsto \mathbf{1}\left(\|\mathbf{x}\| \leq 1\right)$ in $\mathbb{R}^d$, which can be easily approximated to accuracy ϵ using a 3-layer network with $\mathcal{O}(d^2/\epsilon)$ neurons, cannot be approximated to an accuracy higher than $\mathcal{O}(1/d^4)$ using a 2-layer network, unless its width is exponential in d. In fact, we show the same result more generally, for any indicator of an ellipsoid $\mathbf{x} \mapsto \mathbf{1}\left(\|A\mathbf{x}+\mathbf{b}\| \leq r\right)$ (where A is a non-singular matrix and $\mathbf{b}$ is a vector). The proof is based on a reduction from the main result of (Eldan & Shamir, 2016), which shows a separation between 2-layer and 3-layer networks using a more complicated and less natural radial function.
- We prove that *any* L_1 radial function $\mathbf{x} \mapsto f(\|\mathbf{x}\|_1)$, where $\mathbf{x} \in \mathbb{R}^d$ and $f : \mathbb{R} \to \mathbb{R}$ is piecewise-linear, cannot be approximated to accuracy ϵ by a depth 2 ReLU network of width less than

[1]Weizmann Institute of Science, Rehovot, Israel. Correspondence to: Itay Safran <itay.safran@weizmann.ac.il>, Ohad Shamir <ohad.shamir@weizmann.ac.il>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

$\tilde{\Omega}(\min\{1/\epsilon, \exp(\Omega(d))\})$. In contrast, such functions can be represented *exactly* by 3-layer ReLU networks.

- We show that this depth/width trade-off can also be observed experimentally: Specifically, that when using standard backpropagation to learn the indicators of the L_1 and L_2 unit balls, 3-layer nets give significantly better performance compared to 2-layer nets (even if much larger). Our theoretical results indicate that this gap in performance is due to approximation error issues. This experiment also highlights the fact that our separation result is for a natural function that is not just well-approximated by some 3-layer network, but can also be learned well from data using standard methods.

- Finally, we prove that any member of a wide family of non-linear and twice-differentiable functions (including for instance $x \mapsto x^2$ in $[0,1]$), which can be approximated to accuracy ϵ using ReLU networks of depth and width $\mathcal{O}(\text{poly}(\log(1/\epsilon)))$, cannot be approximated to similar accuracy by constant-depth ReLU networks, unless their width is at least $\Omega(\text{poly}(1/\epsilon))$. We note that a similar result appeared online concurrently and independently of ours in (Yarotsky, 2016; Liang & Srikant, 2016), but the setting is a bit different (see related work below for more details).

RELATED WORK

The question of studying the effect of depth in neural network has received considerable attention recently, and studied under various settings. Many of these works consider a somewhat different setting than ours, and hence are not directly comparable. These include networks which are not plain-vanilla ones (e.g. (Cohen et al., 2016; Delalleau & Bengio, 2011; Martens & Medabalimi, 2014)), measuring quantities other than approximation error (e.g. (Bianchini & Scarselli, 2014; Poole et al., 2016)), focusing only on approximation upper bounds (e.g. (Shaham et al., 2016)), or measuring approximation error in terms of L_∞-type bounds, i.e. $\sup_{\mathbf{x}} |f(\mathbf{x}) - \tilde{f}(\mathbf{x}))|$ rather than L_2-type bounds $\mathbb{E}_{\mathbf{x}}(f(\mathbf{x}) - \tilde{f}(\mathbf{x}))^2$ (e.g. (Yarotsky, 2016; Liang & Srikant, 2016)). We note that the latter distinction is important: Although L_∞ bounds are more common in the approximation theory literature, L_2 bounds are more natural in the context of statistical machine learning problems (where we care about the expected loss over some distribution). Moreover, L_2 approximation lower bounds are stronger, in the sense that an L_2 lower bound easily translates to a lower bound on L_∞ lower bound, but not vice versa[1].

[1]To give a trivial example, ReLU networks always express continuous functions, and therefore can never approximate a discontinuous function such as $x \mapsto \mathbf{1}(x \geq 0)$ in an L_∞ sense, yet can easily approximate it in an L_2 sense given any continuous distribution.

A noteworthy paper in the same setting as ours is (Telgarsky, 2016), which proves a separation result between the expressivity of ReLU networks of depth k and depth $o(k/\log(k))$ (for any k). This holds even for one-dimensional functions, where a depth k network is shown to realize a saw-tooth function with $\exp(\mathcal{O}(k))$ oscillations, whereas any network of depth $o(k/\log(k))$ would require a width super-polynomial in k to approximate it by more than a constant. In fact, we ourselves rely on this construction in the proofs of our results in section 5. On the flip side, in our paper we focus on separation in terms of the accuracy or dimension, rather than a parameter k. Moreover, the construction there relies on a highly oscillatory function, with Lipschitz constant exponential in k almost everywhere. In contrast, in our paper we focus on simpler functions, of the type that are likely to be learnable from data using standard methods.

Our separation results in Sec. 5 (for smooth non-linear functions) are closely related to those of (Yarotsky, 2016; Liang & Srikant, 2016), which appeared online concurrently and independently of our work, and the proof ideas are quite similar. However, these papers focused on L_∞ bounds rather than L_2 bounds. Moreover, (Yarotsky, 2016) considers a class of functions different than ours in their positive results, and (Liang & Srikant, 2016) consider networks employing a mix of ReLU and threshold activations, whereas we consider a purely ReLU network.

Another relevant and insightful work is (Poggio et al., 2016), which considers width vs. depth and provide general results on expressibility of functions with a compositional nature. However, the focus there is on worse-case approximation over general classes of functions, rather than separation results in terms of specific functions as we do here, and the details and setting is somewhat orthogonal to ours.

2. Preliminaries

In general, we let bold-faced letters such as $\mathbf{x} = (x_1, \ldots, x_d)$ denote vectors, and capital letters denote matrices or probabilistic events. $\|\cdot\|$ denotes the Euclidean norm, and $\|\cdot\|_1$ the 1-norm. $\mathbf{1}(\cdot)$ denotes the indicator function. We use the standard asymptotic notation $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$ to hide constants, and $\tilde{\mathcal{O}}(\cdot)$ and $\tilde{\Omega}(\cdot)$ to hide constants and factors logarithmic in the problem parameters.

Neural Networks. We consider feed-forward neural networks, computing functions from $\mathbb{R}^d$ to $\mathbb{R}$. The network is composed of layers of neurons, where each neuron computes a function of the form $\mathbf{x} \mapsto \sigma(\mathbf{w}^\top \mathbf{x} + b)$, where $\mathbf{w}$

is a weight vector, b is a bias term and $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is a non-linear activation function, such as the ReLU function $\sigma(z) = [z]_+ = \max\{0, z\}$. Letting $\sigma(W\mathbf{x}+\mathbf{b})$ be a shorthand for $\big(\sigma(\mathbf{w}_1^\top \mathbf{x} + b_1), \ldots, \sigma(\mathbf{w}_n^\top \mathbf{x} + b_n)\big)$, we define a layer of n neurons as $\mathbf{x} \mapsto \sigma(W\mathbf{x} + \mathbf{b})$. By denoting the output of the i^{th} layer as O_i, we can define a network of arbitrary depth recursively by $O_{i+1} = \sigma(W_{i+1}O_i + \mathbf{b}_{i+1})$, where $W_i, \mathbf{b}_i$ represent the matrix of weights and bias of the i^{th} layer, respectively. Following a standard convention for multi-layer networks, the final layer h is a purely linear function with no bias, i.e. $O_h = W_h \cdot O_{h-1}$. We define the *depth* of the network as the number of layers l, and denote the number of neurons n_i in the i^{th} layer as the *size* of the layer. We define the *width* of a network as $\max_{i\in\{1,\ldots,l\}} n_i$. Finally, a *ReLU network* is a neural network where all the non-linear activations are the ReLU function. We use "2-layer" and "3-layer" to denote networks of depth 2 and 3. In particular, in our notation a 2-layer ReLU network has the form

$$\mathbf{x} \mapsto \sum_{i=1}^{n_1} v_i \cdot [\mathbf{w}_i^\top \mathbf{x} + b_i]_+$$

for some parameters $v_1, b_1, \ldots, v_{n_1}, b_{n_1}$ and d-dimensional vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{n_1}$. Similarly, a 3-layer ReLU network has the form

$$\sum_{i=1}^{n_2} u_i \left[\sum_{j=1}^{n_1} v_{i,j} \left[\mathbf{w}_{i,j}^\top \mathbf{x} + b_{i,j} \right]_+ + c_i \right]_+$$

for some parameters $\{u_i, v_{i,j}, b_{i,j}, c_i, \mathbf{w}_{i,j}\}$.

Approximation error. Given some function f on a domain $\mathcal{X}$ endowed with some probability distribution (with density function μ), we define the quality of its approximation by some other function $\tilde{f}$ as $\int_{\mathcal{X}} (f(\mathbf{x}) - \tilde{f}(\mathbf{x}))^2 \mu(\mathbf{x}) d\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim\mu}[(f(\mathbf{x}) - \tilde{f}(\mathbf{x}))^2]$. We refer to this as approximation in the L_2-norm sense. In one of our results (Thm. 6), we also consider approximation in the L_∞-norm sense, defined as $\sup_{\mathbf{x}\in\mathcal{X}} |f(\mathbf{x}) - \tilde{f}(\mathbf{x})|$. Clearly, this upper-bounds the (square root of the) L_2 approximation error defined above, so as discussed in the introduction, lower bounds on the L_2 approximation error (w.r.t. any distribution) are stronger than lower bounds on the L_∞ approximation error.

3. Indicators of L_2 Balls and Ellipsoids

We begin by considering one of the simplest possible function classes on $\mathbb{R}^d$, namely indicators of L_2 balls (and more generally, ellipsoids). The ability to compute such functions is necessary for many useful primitives, for example determining if the distance between two points in Euclidean space is below or above some threshold (either with respect to the Euclidean distance, or a more general Mahalanobis distance). In this section, we show a depth separation result for such functions: Although they can be easily approximated with 3-layer networks, no 2-layer network can approximate it to high accuracy w.r.t. any distribution, unless its width is exponential in the dimension. This is formally stated in the following theorem:

Theorem 1 (Inapproximability with 2-layer networks)**.** *The following holds for some positive universal constants c_1, c_2, c_3, c_4, and any network employing an activation function satisfying Assumptions 1 and 2 in Eldan & Shamir (2016): For any $d > c_1$, and any non-singular matrix $A \in \mathbb{R}^{d\times d}$, $\mathbf{b} \in \mathbb{R}^d$ and $r \in (0, \infty)$, there exists a continuous probability distribution γ on $\mathbb{R}^d$, such that for any function g computed by a 2-layer network of width at most $c_3 \exp(c_4 d)$, and for the function $f(\mathbf{x}) = \mathbf{1}(\|A\mathbf{x} + \mathbf{b}\| \le r)$, we have*

$$\int_{\mathbb{R}^d} \left(f(\mathbf{x}) - g(\mathbf{x})\right)^2 \cdot \gamma(\mathbf{x}) d\mathbf{x} \;\ge\; \frac{c_2}{d^4}.$$

We note that the assumptions from (Eldan & Shamir, 2016) are very mild, and apply to all standard activation functions, including ReLU, sigmoid and threshold. For completeness, the fully stated assumptions are presented in Subsection A.1

The formal proof of Thm. 1 (provided below) is based on a reduction from the main result of (Eldan & Shamir, 2016), which shows the existence of a certain radial function (depending on the input $\mathbf{x}$ only through its norm) and a probability distribution which cannot be expressed by a 2-layer network, whose width is less than exponential in the dimension d to more than constant accuracy. A closer look at the proof reveals that this function (denoted as $\tilde{g}$) can be expressed as a sum of $\Theta(d^2)$ indicators of L_2 balls of various radii. We argue that if we could have accurately approximated a given L_2 ball indicator with respect to all distributions, then we could have approximated all the indicators whose sum add up to $\tilde{g}$, and hence reach a contradiction. By a linear transformation argument, we show the same contradiction would have occured if we could have approximated the indicators of an non-degenerate ellipse with respect to any distribution. The formal proof is provided below:

Proof of Thm. 1. Assume by contradiction that for f as described in the theorem, and for any distribution γ, there exists a 2-layer network $\tilde{f}_\gamma$ of width at most $c_3 \exp(c_4 d)$, such that

$$\int_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) - \tilde{f}_\gamma(\mathbf{x})\right)^2 \gamma(\mathbf{x}) d\mathbf{x} \;\le\; \epsilon \;\le \frac{c_2}{d^4}.$$

Let $\hat{A}$ and $\hat{\mathbf{b}}$ be a $d \times d$ non-singular matrix and vector respectively, to be determined later. We begin by performing a change of variables, $\mathbf{y} = \hat{A}\mathbf{x} + \hat{\mathbf{b}} \iff \mathbf{x} = \hat{A}^{-1}(\mathbf{y} - \hat{\mathbf{b}})$,

$d\mathbf{x} = \left|\det\left(\hat{A}^{-1}\right)\right| \cdot d\mathbf{y}$, which yields

$$\int_{\mathbf{y}\in\mathbb{R}^d} \left(f\left(\hat{A}^{-1}\left(\mathbf{y}-\hat{\mathbf{b}}\right)\right) - \tilde{f}_\gamma\left(\hat{A}^{-1}\left(\mathbf{y}-\hat{\mathbf{b}}\right)\right)\right)^2 \cdot \gamma\left(\hat{A}^{-1}\left(\mathbf{y}-\hat{\mathbf{b}}\right)\right) \cdot \left|\det\left(\hat{A}^{-1}\right)\right| \cdot d\mathbf{y} \;\leq\; \epsilon. \quad (1)$$

In particular, let us choose the distribution γ defined as $\gamma(\mathbf{z}) = |\det(\hat{A})| \cdot \mu(\hat{A}\mathbf{z}+\hat{\mathbf{b}})$, where μ is the (continuous) distribution used in the main result of (Eldan & Shamir, 2016) (note that γ is indeed a distribution, since $\int_{\mathbf{z}} \gamma(\mathbf{z}) = (\det(\hat{A})) \int_{\mathbf{z}} \mu(\hat{A}\mathbf{z}+\hat{\mathbf{b}})d\mathbf{z}$, which by the change of variables $\mathbf{x} = \hat{A}\mathbf{z}+\hat{\mathbf{b}}$, $d\mathbf{x} = |\det(\hat{A})|d\mathbf{z}$ equals $\int_{\mathbf{x}} \mu(\mathbf{x})d\mathbf{x} = 1$). Plugging the definition of γ in Eq. (1), and using the fact that $|\det(\hat{A}^{-1})| \cdot |\det(\hat{A})| = 1$, we get

$$\int_{\mathbf{y}\in\mathbb{R}^d} \left(f\left(\hat{A}^{-1}\left(\mathbf{y}-\hat{\mathbf{b}}\right)\right) - \tilde{f}_\gamma\left(\hat{A}^{-1}\left(\mathbf{y}-\hat{\mathbf{b}}\right)\right)\right)^2 \cdot \mu(\mathbf{y})\, d\mathbf{y} \;\leq\; \epsilon. \quad (2)$$

Letting $z > 0$ be an arbitrary parameter, we now pick $\hat{A} = \frac{z}{r}A$ and $\hat{\mathbf{b}} = \frac{z}{r}\mathbf{b}$. Recalling the definition of f as $\mathbf{x} \mapsto \mathbf{1}(\|A\mathbf{x}+\mathbf{b}\| \leq r)$, we get that

$$\int_{\mathbf{y}\in\mathbb{R}^d} \left(\mathbf{1}(\|\mathbf{y}\| \leq z) - \tilde{f}_\gamma\left(\frac{r}{z}A^{-1}\left(\mathbf{y}-\frac{z}{r}\mathbf{b}\right)\right)\right)^2 \cdot \mu(\mathbf{y})\, d\mathbf{y} \;\leq\; \epsilon. \quad (3)$$

Note that $\tilde{f}_\gamma\left(\frac{r}{z}A^{-1}\left(y-\frac{z}{r}\mathbf{b}\right)\right)$ expresses a 2-layer network composed with a linear transformation of the input, and hence can be expressed in turn by a 2-layer network (as we can absorb the linear transformation into the parameters of each neuron in the first layer). Therefore, letting $\|f\|_{L_2(\mu)} = \sqrt{\int_{\mathbf{y}} f^2(\mathbf{y})d\mathbf{y}}$ denote the norm in $L_2(\mu)$ function space, we showed the following: For any $z > 0$, there exists a 2-layer network $\tilde{f}_z$ such that

$$\left\|\left(\mathbf{1}(\|\cdot\| \leq z) - \tilde{f}_z(\cdot)\right)\right\|_{L_2(\mu)} \;\leq\; \sqrt{\epsilon}. \quad (4)$$

With this key result in hand, we now turn to complete the proof. We consider the function $\tilde{g}$ from (Eldan & Shamir, 2016), for which it was proven that no 2-layer network can approximate it w.r.t. μ to better than constant accuracy, unless its width is exponential in the dimension d. In particular $\tilde{g}$ can be written as

$$\tilde{g}(\mathbf{x}) = \sum_{i=1}^{n} \epsilon_i \cdot \mathbf{1}(\|\mathbf{x}\| \in [a_i, b_i]),$$

where $[a_i, b_i]$ are disjoint intervals, $\epsilon_i \in \{-1, +1\}$, and $n = \Theta(d^2)$ where d is the dimension. Since $\tilde{g}$ can also be written as

$$\sum_{i=1}^{n} \epsilon_i \left(\mathbf{1}(\|\mathbf{x}\| \leq b_i) - \mathbf{1}(\|\mathbf{x}\| \leq a_i)\right),$$

we get by Eq. (4) and the triangle inequality that

$$\begin{aligned}
&\left\|\tilde{g}(\cdot) - \sum_{i=1}^{n} \epsilon_i \cdot (\tilde{f}_{b_i}(\cdot) - \tilde{f}_{a_i}(\cdot)\right\|_{L_2(\mu)} \\
&\leq \sum_{i=1}^{n} |\epsilon_i| \left(\left\|\left(\mathbf{1}(\|\cdot\| \leq b_i) - \tilde{f}_{b_i}\right)\right\|_{L_2(\mu)} + \left\|\mathbf{1}(\|\cdot\| \leq a_i) - \tilde{f}_{a_i}(\cdot)\right\|_{L_2(\mu)}\right) \\
&\leq\; 2n\sqrt{\epsilon}.
\end{aligned}$$

However, since a linear combination of $2n$ 2-layer neural networks of width at most w is still a 2-layer network, of width at most $2nw$, we get that $\sum_{i=1}^{n} \epsilon_i \cdot (\tilde{f}_{b_i}(\cdot) - \tilde{f}_{a_i}(\cdot))$ is a 2-layer network, of width at most $\Theta(d^2) \cdot c_3 \exp(c_4 d)$, which approximates $\tilde{g}$ to an accuracy of less than $2n\sqrt{\epsilon} = \Theta(d^2) \cdot \sqrt{c_2/d^4} = \Theta(1) \cdot \sqrt{c_2}$. Hence, by picking c_2, c_3, c_4 sufficiently small, we get a contradiction to the result of (Eldan & Shamir, 2016), that no 2-layer network of width smaller than $c\exp(cd)$ (for some constant c) can approximate $\tilde{g}$ to more than constant accuracy, for a sufficiently large dimension d. □

To complement Thm. 1, we also show that such indicator functions can be easily approximated with 3-layer networks. The argument is quite simple: Using an activation such as ReLU or Sigmoid, we can use one layer to approximate any Lipschitz continuous function on any bounded interval, and in particular $x \mapsto x^2$. Given a vector $\mathbf{x} \in \mathbb{R}^d$, we can apply this construction on each coordinate x_i seperately, hence approximating $\mathbf{x} \mapsto \|\mathbf{x}\|^2 = \sum_{i=1}^{d} x_i^2$. Similarly, we can approximate $\mathbf{x} \mapsto \|A\mathbf{x}+\mathbf{b}\|$ for arbitrary fixed matrices A and vectors $\mathbf{b}$. Finally, with a 3-layer network, we can use the second layer to compute a continuous approximation to the threshold function $z \mapsto \mathbf{1}(z \leq r)$. Composing these two layers, we get an arbitrarily good approximation to the function $\mathbf{x} \mapsto \mathbf{1}(\|A\mathbf{x}+\mathbf{b}\| \leq r)$ w.r.t. any continuous distribution, with the network size scaling polynomially with the dimension d and the required accuracy. In the theorem below, we formalize this intuition, where for simplicity we focus on approximating the indicator of the unit ball:

Theorem 2 (Approximability with 3-layer networks)**.** *Given $\delta > 0$, for any activation function σ satisfying Assumption 1 in Eldan & Shamir (2016) and any continuous probability distribution μ on $\mathbb{R}^d$, there exists a constant c_σ dependent only on σ, and a function g expressible by a 3-layer network of width at most $\max\left\{8c_\sigma d^2/\delta, c_\sigma\sqrt{1/2\delta}\right\}$, such that the following holds:*

$$\int_{\mathbb{R}^d} \left(g(\mathbf{x}) - \mathbf{1}(\|\mathbf{x}\|_2 \leq 1)\right)^2 \mu(\mathbf{x})\, d\mathbf{x} \;\leq\; \delta,$$

where c_σ is a constant depending solely on σ.

The proof of the theorem appears in the supplementary material

3.1. An Experiment

In this subsection, we empirically demonstrate that indicator functions of L_2 balls are indeed easier to learn with a 3-layer network, compared to a 2-layer network (even if the 2-layer network is significantly larger). This indicates that the depth/width trade-off for indicators of balls, predicted by our theory, can indeed be observed experimentally. Moreover, it highlights the fact that our separation result is for simple natural functions, that can be learned reasonably well from data using standard methods.

For our experiment, we sampled $5 \cdot 10^5$ data instances in $\mathbb{R}^{100}$, with a direction chosen uniformly at random and a norm drawn uniformly at random from the interval $[0, 2]$. To each instance, we associated a target value computed according to the target function $f(\mathbf{x}) = \mathbf{1}\left(\|\mathbf{x}\|_2 \leq 1\right)$. Another $5 \cdot 10^4$ examples were generated in a similar manner and used as a validation set.

We trained 5 ReLU networks on this dataset:

- One 3-layer network, with a first hidden layer of size 100, a second hidden layer of size 20, and a linear output neuron.
- Four 2-layer networks, with hidden layer of sizes $100, 200, 400$ and 800, and a linear output neuron.

Training was performed with backpropagation, using the TensorFlow library. We used the squared loss $\ell(y, y') = (y - y')^2$ and batches of size 100. For all networks, we chose a momentum parameter of 0.95, and a learning rate starting at 0.1, decaying by a multiplicative factor of 0.95 every 1000 batches, and stopping at 10^{-4}.

The results are presented in Fig. 1. As can be clearly seen, the 3-layer network achieves significantly better performance than the 2-layer networks. This is true even though some of these networks are significantly larger and with more parameters (for example, the 2-layer, width 800 network has ˜80K parameters, vs. ˜10K parameters for the 3-layer network). This gap in performance is the exact opposite of what might be expected based on parameter counting alone. Moreover, increasing the width of the 2-layer networks exhibits diminishing returns: The performance improvement in doubling the width from 100 to 200 is much larger than doubling the width from 200 to 400 or 400 to 800. This indicates that one would need a much larger 2-layer network to match the 3-layer, width 100 network's performance. Thus, we conclude that the network's depth indeed plays a crucial role, and that 3-layer networks are inherently more suitable to express indicator functions of the type we studied.

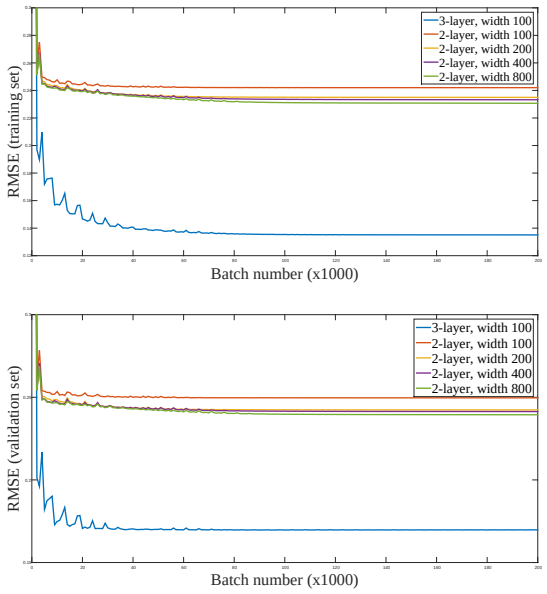

Figure 1. The experiment results, depicting the network's root mean square error over the training set (top) and validation set (bottom), as a function of the number of batches processed. Best viewed in color.

4. L_1 Radial Functions; ReLU Networks

Having considered functions depending on the L_2 norm, we now turn to consider functions depending on the L_1 norm. Focusing on ReLU networks, we will show a certain separation result holding for *any* non-linear function, which depends on the input $\mathbf{x}$ only via its 1-norm $\|\mathbf{x}\|_1$.

Theorem 3. *Let $f : [0, \infty) \mapsto \mathbb{R}$ be a function such that for some $r, \delta > 0$ and $\epsilon \in (0, 1/2)$,*

$$\inf_{a,b \in \mathbb{R}} \mathbb{E}_{x \text{ uniform on } [r,(1+\epsilon)r]}[(f(x) - (ax - b))^2] > \delta \ .$$

Then there exists a distribution γ over $\{\mathbf{x} : \|\mathbf{x}\|_1 \leq (1 + \epsilon)r\}$, such that if a 2-layer ReLU network $F(\mathbf{x})$ satisfies

$$\int_{\mathbf{x}} \left(f(\|\mathbf{x}\|_1) - F(\mathbf{x})\right)^2 \gamma(\mathbf{x}) d\mathbf{x} \ \leq \ \delta/2,$$

then its width must be at least $\tilde{\Omega}(\min\{1/\epsilon, \exp(\Omega(d))\})$ (where the $\tilde{\Omega}$ notation hides constants and factors logarithmic in ϵ, d).

The proof appears in the supplementary material. We note that δ controls how 'linearly inapproximable' is f in a narrow interval (of width ϵ) around r, and that δ is generally dependent on ϵ. To give a concrete example, suppose that $f(z) = [z - 1]_+$, which cannot be approximated by a linear function to an accuracy better than $\mathcal{O}(\epsilon^2)$ in an ϵ-neighborhood of 1. By taking $r = 1 - \frac{\epsilon}{2}$ and $\delta = \mathcal{O}(\epsilon^2)$, we get that no 2-layer network can approximate the function

$[\|\mathbf{x}\|_1 - 1]_+$ (at least with respect to some distribution), unless its width is $\tilde{\Omega}(\min\{1/\epsilon, \exp(\Omega(d))\})$. On the flip side, $f(\|\mathbf{x}\|_1)$ can be expressed *exactly* by a 3-layer, width $2d$ ReLU network: $\mathbf{x} \mapsto [\sum_{i=1}^{d}([x_i]_+ + [-x_i]_+) - 1]_+$, where the output neuron is simply the identity function. The same argument would work for any piecewise-linear f. More generally, the same kind of argument would work for any function f exhibiting a non-linear behavior at some points: Such functions can be well-approximated by 3-layer networks (by approximating f with a piecewise-linear function), yet any approximating 2-layer network will have a lower bound on its size as specified in the theorem.

Intuitively, the proof relies on showing that any good 2-layer approximation of $f(\|\mathbf{x}\|_1)$ must capture the non-linear behavior of f close to "most" points $\mathbf{x}$ satisfying $\|\mathbf{x}\|_1 \approx r$. However, a 2-layer ReLU network $\mathbf{x} \mapsto \sum_{j=1}^{N} a_j [\langle \mathbf{w}_j, \mathbf{x}\rangle + b_j]_+$ is piecewise linear, with non-linearities only at the union of the N hyperplanes $\cup_j \{\mathbf{x} : \langle \mathbf{w}_j, \mathbf{x}\rangle + b_j = 0\}$. This implies that "most" points $\mathbf{x}$ s.t. $\|\mathbf{x}\|_1 \approx r$ must be ϵ-close to a hyperplane $\{\mathbf{x} : \langle \mathbf{w}_j, \mathbf{x}\rangle + b_j = 0\}$. However, the geometry of the L_1 ball $\{\mathbf{x} : \|\mathbf{x}\| = r\}$ is such that the ϵ neighborhood of any single hyperplane can only cover a "small" portion of that ball, yet we need to cover most of the L_1 ball. Using this and an appropriate construction, we show that required number of hyperplanes is at least $1/\epsilon$, as long as $\epsilon > \exp(-\mathcal{O}(d))$ (and if ϵ is smaller than that, we can simply use one neuron/hyperplane for each of the 2^d facets of the L_1 ball, and get a covering using 2^d neurons/hyperplanes). The formal proof appears in the supplementary material.

We note that the bound in Thm. 3 is of a weaker nature than the bound in the previous section, in that the lower bound is only polynomial rather than exponential (albeit w.r.t. different problem parameters: ϵ vs. d). Nevertheless, we believe this does point out that L_1 balls also pose a geometric difficulty for 2-layer networks, and conjecture that our lower bound can be considerably improved: Indeed, at the moment we do not know how to approximate a function such as $\mathbf{x} \mapsto [\|\mathbf{x}\|_1 - 1]_+$ with 2-layer networks to better than constant accuracy, using less than $\Omega(2^d)$ neurons.

Finally, we performed an experiment similar to the one presented in Subsection 3.1, where we verified that the bounds we derived are indeed reflected in differences in empirical performance, when training 2-layer nets versus 3-layer nets. The reader is referred to Sec. B for the full details of the experiment and its results.

5. C^2 Nonlinear Functions; ReLU Networks

In this section, we establish a depth separation result for approximating continuously twice-differentiable (C^2) functions using ReLU neural networks. Unlike the previous results in this paper, the separation is for depths which can be larger than 3, depending on the required approximation error. Also, the results will all be with respect to the uniform distribution μ_d over $[0,1]^d$. As mentioned earlier, the results and techniques in this section are closely related to the independent results of (Yarotsky, 2016; Liang & Srikant, 2016), but our emphasis is on L_2 rather than L_∞ approximation bounds, and we focus on somewhat different network architectures and function classes.

Clearly, not all C^2 functions are difficult to approximate (e.g. a linear function can be expressed exactly with a 2-layer network). Instead, we consider functions which have a certain degree of non-linearity, in the sense that its Hessians are non-zero along some direction, on a significant portion of the domain. Formally, we make the following definition:

Definition 1. *Let μ_d denote the uniform distribution on $[0,1]^d$. For a function $f : [0,1]^d \to \mathbb{R}$ and some $\lambda > 0$, denote*

$$\sigma_\lambda(f) = \sup_{\mathbf{v}\in\mathbb{S}^{d-1},\, U\in\mathcal{U} \text{ s.t. } \mathbf{v}^\top H(f)(\mathbf{x})\mathbf{v}\geq\lambda\ \forall \mathbf{x}\in U} \mu_d(U),$$

where $\mathbb{S}^{d-1} = \{\mathbf{x} : \|\mathbf{x}\|_2 = 1\}$ is the d-dimensional unit hypersphere, and $\mathcal{U}$ is the set of all connected and measurable subsets of $[0,1]^d$.

In words, $\sigma_\lambda(f)$ is the measure (w.r.t. the uniform distribution on $[0,1]^d$) of the largest connected set in the domain of f, where at any point, f has curvature at least λ along some fixed direction $\mathbf{v}$. The "prototypical" functions f we are interested in is when $\sigma_\lambda(f)$ is lower bounded by a constant (e.g. it is 1 if f is strongly convex). We stress that our results in this section will hold equally well by considering the condition $\mathbf{v}^\top H(f)(\mathbf{x})\mathbf{v} \leq -\lambda$ as well, however for the sake of simplicity we focus on the former condition appearing in Def. 1. Our goal is to show a depth separation result *inidividually* for any such function (that is, for any such function, there is a gap in the attainable error between deeper and shallower networks, even if the shallow network is considerably larger).

As usual, we start with an inapproximability result. Specifically, we prove the following lower bound on the attainable approximation error of f, using a ReLU neural network of a given depth and width:

Theorem 4. *For any C^2 function $f : [0,1]^d \to \mathbb{R}$, any $\lambda > 0$, and any function g on $[0,1]^d$ expressible by a ReLU network of depth l and maximal width m, it holds that*

$$\int_{[0,1]^d} (f(\mathbf{x}) - g(\mathbf{x})^2 \mu_d(\mathbf{x})\, d\mathbf{x} \geq \frac{c\cdot\lambda^2\cdot\sigma_\lambda^5}{(2m)^{4l}},$$

where $c > 0$ is a universal constant.

The theorem conveys a key tradeoff between depth and width when approximating a C^2 function using ReLU networks: The error cannot decay faster than polynomially in the width m, yet the bound deteriorates exponentially in the depth l. As we show later on, this deterioration does not stem from the looseness in the bound: For well-behaved f, it is indeed possible to construct ReLU networks, where the approximation error decays exponentially with depth.

The proof of Thm. 4 appears in the supplementary material, and is based on a series of intermediate results. First, we show that any strictly curved function (in a sense similar to Definition 1) cannot be well-approximated in an L_2 sense by piecewise linear functions, unless the number of linear regions is large. To that end, we first establish some necessary tools based on Legendre polynomials. We then prove a result specific to the one-dimensional case, including an explicit lower bound if the target function is quadratic (Thm. 9) or strongly convex or concave (Thm. 10). We then expand the construction to get an error lower bound in general dimension d, depending on the number of linear regions in the approximating piecewise-linear function. Finally, we note that any ReLU network induces a piecewise-linear function, and bound the number of linear regions induced by a ReLU network of a given width and depth (using a lemma borrowed from (Telgarsky, 2016)). Combining this with the previous lower bound yields Thm. 4.

We now turn to complement this lower bound with an approximability result, showing that with more depth, a wide family of functions to which Thm. 4 applies *can* be approximated with exponentially high accuracy. Specifically, we consider functions which can be approximated using a moderate number of multiplications and additions, where the values of intermediate computations are bounded (for example, a special case is any function approximable by a moderately-sized Boolean circuit, or a polynomial).

The key result to show this is the following, which implies that the multiplication of two (bounded-size) numbers can be approximated by a ReLU network, with error decaying exponentially with depth:

Theorem 5. *Let $f : [-M, M]^2 \to \mathbb{R}$, $f(x, y) = x \cdot y$ and let $\epsilon > 0$ be arbitrary. Then exists a ReLU neural network g of width $4\left\lceil \log\left(\frac{M}{\epsilon}\right)\right\rceil + 13$ and depth $\left\lceil 2\log\left(\frac{M}{\epsilon}\right)\right\rceil + 9$ satisfying*

$$\sup_{(x,y)\in[-M,M]^2} |f(x, y) - g(x, y)| \leq \epsilon.$$

The idea of the construction is that depth allows us to compute highly-oscillating functions, which can extract high-order bits from the binary representation of the inputs. Given these bits, one can compute the product by a procedure resembling long multiplication, as shown in Fig. 2,

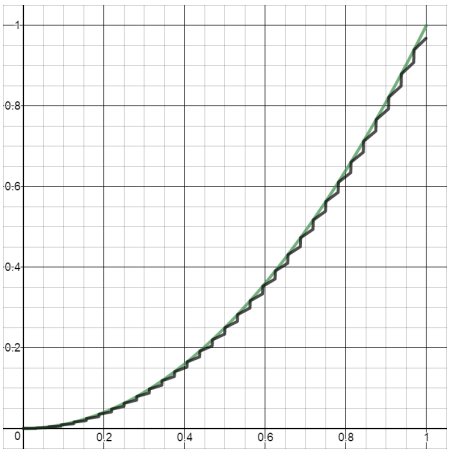

Figure 2. ReLU approximation of the function $x \mapsto x^2$ obtained by extracting 5 bits. The number of linear segments grows exponentially with the number of bits and the approximating network size.

and formally proven as follows:

Proof of Thm. 5. We begin by observing that by using a simple linear change of variables on x, we may assume without loss of generality that $x \in [0, 1]$, as we can just rescale x to the interval $[0, 1]$, and then map it back to its original domain $[-M, M]$, where the error will multiply by a factor of $2M$. Then by requiring accuracy $\frac{\epsilon}{2M}$ instead of ϵ, the result will follow.

The key behind the proof is that performing bit-wise operations on the first k bits of $x \in [0, 1]$ yields an estimation of the product to accuracy $2^{1-k}M$. Let $x = \sum_{i=1}^{\infty} 2^{-i}x_i$ be the binary representation of x where x_i is the i^{th} bit of x, then

$$\begin{aligned} x \cdot y &= \sum_{i=1}^{\infty} 2^{-i}x_i \cdot y \\ &= \sum_{i=1}^{k} 2^{-i}x_i \cdot y + \sum_{i=k+1}^{\infty} 2^{-i}x_i \cdot y. \end{aligned} \tag{5}$$

But since

$$\left|\sum_{i=k+1}^{\infty} 2^{-i}x_i \cdot y\right| \leq \left|\sum_{i=k+1}^{\infty} 2^{-i} \cdot y\right| = 2^{-k}|y| \leq 2^{1-k}M,$$

Eq. (5) implies

$$\left|x \cdot y - \sum_{i=1}^{k} 2^{-i}x_i \cdot y\right| \leq 2^{1-k}M.$$

Requiring that $2^{2-k}M \leq \frac{\epsilon}{2M}$, it suffices to show the existence of a network which approximates the function $\sum_{i=1}^{k} 2^{-i}x_i \cdot y$ to accuracy $\frac{\epsilon}{2}$, where $k = 2\left\lceil \log\left(\frac{8M}{\epsilon}\right)\right\rceil$.

This way both approximations will be at most $\frac{\epsilon}{2}$, resulting in the desired accuracy of ϵ.

Before specifying the architecture which extracts the i^{th} bit of x, we first describe the last 2 layers of the network. Let the penultimate layer comprise of k neurons, each receiving both y and x_i as input, and having the set of weights $\left(2^{-i}, 1, -1\right)$. Thus, the output of the i^{th} neuron in the penultimate layer is

$$\left[2^{-i}y + x_i - 1\right]_+ = 2^{-i}x_i y.$$

Let the final single output neuron have the set of weights $(1, \ldots, 1, 0) \in \mathbb{R}^{k+1}$, this way, the output of the network will be $\sum_{i=1}^{k} 2^{-i}x_i \cdot y$ as required.

We now specify the architecture which extracts the first most significant k bits of x. In Telgarsky (2016), the author demonstrates how the composition of the function

$$\varphi(x) = [2x]_+ - [4x - 2]_+$$

with itself i times, φ^i, yields a highly oscillatory triangle wave function in the domain $[0, 1]$. Furthermore, we observe that $\varphi(x) = 0 \; \forall x \leq 0$, and thus $\varphi^i(x) = 0 \; \forall x \leq 0$. Now, a linear shift of the input of φ^i by 2^{-i-1}, and composing the output with

$$\sigma_\delta(x) = \left[\frac{1}{2\delta}x - \frac{1}{4\delta} + \frac{1}{2}\right]_+ - \left[\frac{1}{2\delta}x - \frac{1}{4\delta} - \frac{1}{2}\right]_+,$$

which converges to $\mathbf{1}_{[x \geq 0.5]}(x)$ as $\delta \to 0$, results in an approximation of $x \mapsto x_i$: $\sigma_\delta\left(\varphi^i\left(x - 2^{-i-1}\right)\right)$. We stress that choosing δ such that the network approximates the bitwise product to accuracy $\frac{\epsilon}{2}$ will require δ to be of magnitude $\frac{1}{\epsilon}$, but this poses no problem as representing such a number requires $\log\left(\frac{1}{\epsilon}\right)$ bits, which is also the magnitude of the size of the network, as suggested by the following analysis.

Next, we compute the size of the network required to implement the above approximation. To compute φ only two neurons are required, therefore φ^i can be computed using i layers with 2 neurons in each, and finally composing this with σ_δ requires a subsequent layer with 2 more neurons. To implement the i^{th} bit extractor we therefore require a network of size $2 \times (i + 1)$. Using dummy neurons to propagate the i^{th} bit for $i < k$, the architecture extracting the k most significant bits of x will be of size $2k \times (k + 1)$. Adding the final component performing the multiplication estimation will require 2 more layers of width k and 1 respectively, and an increase of the width by 1 to propagate y to the penultimate layer, resulting in a network of size $(2k + 1) \times (k + 1)$. □

Thm. 5 shows that multiplication can be performed very accurately by deep networks. Moreover, additions can be computed by ReLU networks exactly, using only a single layer with 4 neurons: Let $\alpha, \beta \in \mathbb{R}$ be arbitrary, then $(x, y) \mapsto \alpha \cdot x + \beta \cdot y$ is given in terms of ReLU summation by

$$\alpha[x]_+ - \alpha[-x]_+ + \beta[y]_+ - \beta[-y]_+.$$

Repeating these arguments, we see that any function which can be approximated by a bounded number of operations involving additions and multiplications, can also be approximated well by moderately-sized networks. This is formalized in the following theorem, which provides an approximation error upper bound (in the L_∞ sense, which is stronger than L_2 for upper bounds):

Theorem 6. *Let $\mathcal{F}_{t,M,\epsilon}$ be the family of functions on the domain $[0, 1]^d$ with the property that $f \in \mathcal{F}_{t,M,\epsilon}$ is approximable to accuracy ϵ with respect to the infinity norm, using at most t operations involving weighted addition, $(x, y) \mapsto \alpha \cdot x + \beta \cdot y$, where $\alpha, \beta \in \mathbb{R}$ are fixed; and multiplication, $(x, y) \mapsto x \cdot y$, where each intermediate computation stage is bounded in the interval $[-M, M]$. Then there exists a universal constant c, and a ReLU network g of width and depth at most $c\left(t \log\left(\frac{1}{\epsilon}\right) + t^2 \log(M)\right)$, such that*

$$\sup_{\mathbf{x} \in [0,1]^d} |f(\mathbf{x}) - g(\mathbf{x})| \leq 2\epsilon.$$

As discussed in Sec. 2, this type of L_∞ approximation bound implies an L_2 approximation bound with respect to any distribution. The proof of the theorem appears in Sec. A.

Combining Thm. 4 and Thm. 6, we can state the following corollary, which formally shows how depth can be exponentially more valuable than width as a function of the target accuracy ϵ:

Corollary 1. *Suppose $f \in C^2 \cap \mathcal{F}_{t(\epsilon),M(\epsilon),\epsilon}$, where $t(\epsilon) = \mathcal{O}(\text{poly}(\log(1/\epsilon)))$ and $M(\epsilon) = \mathcal{O}(\text{poly}(1/\epsilon))$. Then approximating f to accuracy ϵ in the L_2 norm using a fixed depth ReLU network requires width at least* $\text{poly}(1/\epsilon)$*, whereas there exists a ReLU network of depth and width at most $p(\log(1/\epsilon))$ which approximates f to accuracy ϵ in the infinity norm, where p is a polynomial depending solely on f.*

Proof. The lower bound follows immediately from Thm. 4. For the upper bound, observe that Thm. 6 implies an ϵ approximation by a network of width and depth at most

$$c\left(t(\epsilon/2)\log(2/\epsilon) + (t(\epsilon/2))^2 \log(M(\epsilon/2))\right),$$

which by the assumption of Corollary 1, can be bounded by $p(\log(1/\epsilon))$ for some polynomial p which depends solely on f. □

Acknowledgements

This research is supported in part by an FP7 Marie Curie CIG grant, Israel Science Foundation grant 425/13, and the Intel ICRI-CI Institute. We would like to thank Shai Shalev-Shwartz for some illuminating discussions, and Eran Amar for his valuable help with the experiments.

References

Bianchini, M. and Scarselli, F. On the complexity of shallow and deep neural network classifiers. In *ESANN*, 2014.

Cohen, Nadav, Sharir, Or, and Shashua, Amnon. On the expressive power of deep learning: A tensor analysis. In *29th Annual Conference on Learning Theory*, pp. 698–728, 2016.

Cybenko, George. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Delalleau, O. and Bengio, Y. Shallow vs. deep sum-product networks. In *NIPS*, pp. 666–674, 2011.

Eldan, Ronen and Shamir, Ohad. The power of depth for feedforward neural networks. In *29th Annual Conference on Learning Theory*, pp. 907–940, 2016.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Hornik, Kurt. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Liang, Shiyu and Srikant, R. Why deep neural networks? *arXiv preprint arXiv:1610.04161*, 2016.

Martens, J. and Medabalimi, V. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

Poggio, Tomaso, Mhaskar, Hrushikesh, Rosasco, Lorenzo, Miranda, Brando, and Liao, Qianli. Why and when can deep–but not shallow–networks avoid the curse of dimensionality: a review. *arXiv preprint arXiv:1611.00740*, 2016.

Poole, Ben, Lahiri, Subhaneil, Raghu, Maithreyi, Sohl-Dickstein, Jascha, and Ganguli, Surya. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, pp. 3360–3368, 2016.

Shaham, Uri, Cloninger, Alexander, and Coifman, Ronald R. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 2016.

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

Yarotsky, Dmitry. Error bounds for approximations with deep relu networks. *arXiv preprint arXiv:1610.01145*, 2016.