Complexity of Linear Regions in Deep Networks

Boris Hanin [*1] **David Rolnick** [*2]

Abstract

It is well-known that the expressivity of a neural network depends on its architecture, with deeper networks expressing more complex functions. In the case of networks that compute piecewise linear functions, such as those with ReLU activation, the number of distinct linear regions is a natural measure of expressivity. It is possible to construct networks with merely a single region, or for which the number of linear regions grows exponentially with depth; it is not clear where within this range most networks fall in practice, either before or after training. In this paper, we provide a mathematical framework to count the number of linear regions of a piecewise linear network and measure the volume of the boundaries between these regions. In particular, we prove that for networks at initialization, the average number of regions along any one-dimensional subspace grows linearly in the total number of neurons, far below the exponential upper bound. We also find that the average distance to the nearest region boundary at initialization scales like the inverse of the number of neurons. Our theory suggests that, even after training, the number of linear regions is far below exponential, an intuition that matches our empirical observations. We conclude that the practical expressivity of neural networks is likely far below that of the theoretical maximum, and that this gap can be quantified.

1. Introduction

A growing field of theory has sought to explain the broad success of deep neural networks via a mathematical characterization of the ability of these networks to approximate different functions of input data. Many such works consider the *expressivity* of neural networks, showing that certain functions are more efficiently expressible by deep architectures than by shallow ones (e.g. Bianchini & Scarselli (2014); Montufar et al. (2014); Telgarsky (2015); Lin et al. (2017); Rolnick & Tegmark (2018)). It has, however, also been noted that many expressible functions are not efficiently *learnable*, at least by gradient descent (Shalev-Shwartz et al., 2018). More generally, the *typical* behavior of a network used in practice, the *practical expressivity*, may be very different from what is theoretically attainable. To adequately explain the power of deep learning, it is necessary to consider networks with parameters as they will naturally occur before, during, and after training.

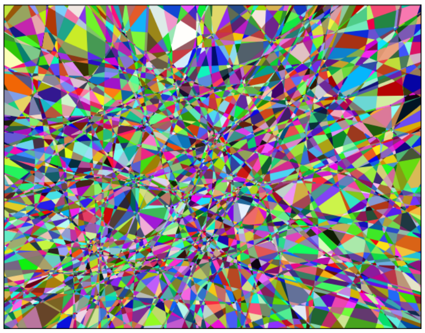


Figure 1. How many linear regions? This figure shows a two-dimensional slice through the 784-dimensional input space of vectorized MNIST, as represented by a fully-connected ReLU network with three hidden layers of width 64 each. Colors denote different linear regions of the piecewise linear network.

Networks with a piecewise linear activation (e.g. ReLU, hard $\tanh$) compute piecewise linear functions for which input space is divided into pieces, with the network computing a single linear function on each piece (see Figures 1-4). Figure 2 shows how the complexity of these pieces, which we refer to as *linear regions*, changes in a deep ReLU net with two-dimensional inputs. Each neuron in the first layer splits the input space into two pieces along a hyperplane, fitting a different linear function to each of the pieces. Subsequent layers split the regions of the preceding layers. The local density of linear regions serves as a convenient proxy for the local complexity or smoothness of the network, with the ability to interpolate a complex data distribution seeming to require fitting many relatively small regions. The topic of

[*] Equal contribution [1] Department of Mathematics, Texas A&M University and Facebook AI Research, New York [2] University of Pennsylvania. Correspondence to: Boris Hanin <bhanin@tamu.edu>, David Rolnick <drolnick@seas.upenn.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

counting linear regions is taken up by a number of authors (Telgarsky, 2015; Montufar et al., 2014; Serra et al., 2018; Raghu et al., 2017).

A worst case estimate is that every neuron in each new layer splits each of the regions present at the previous layer, giving a number of regions exponential in the depth. Indeed this is possible, as examined extensively e.g. in Montufar et al. (2014). An example of Telgarsky (2015) shows that a sawtooth function with 2^n teeth can be expressed exactly using only $3n + 4$ neurons, as shown in Figure 3. However, even slightly perturbing this network (by adding noise to the weights and biases) ruins this beautiful structure and severely reduces the number of linear pieces, raising the question of whether typical neural networks actually achieve the theoretical bounds for numbers of linear regions.

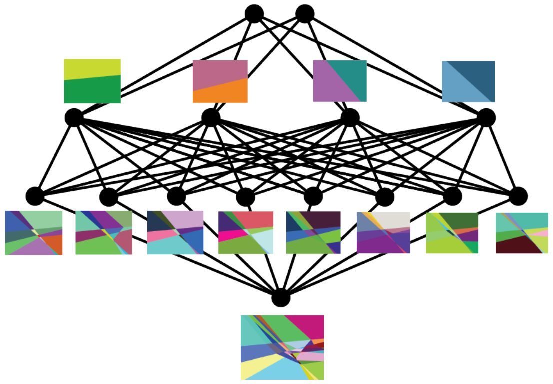
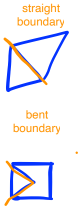


Figure 2. Evolution of linear regions within a ReLU network for 2-dimensional input. Each neuron in the first layer defines a linear boundary that partitions the input space into two regions. Neurons in the second layer combine and split these linear boundaries into higher level patterns of regions, and so on. Ultimately, the input space is partitioned into a number of regions, on each of which the neural network is given by a (different) linear function. During training, both the partition into regions and the linear functions on them are learned.

Figure 1 also invites measures of complexity for piecewise linear networks beyond region counting. The boundary between two linear regions can be straight or can be bent in complex ways, for example, suggesting the *volume of the boundary* between linear regions as complexity measure for the resulting partition of input space. In the 2D example of Figure 1, this corresponds to computing perimeters of the linear pieces. As we detail below, this measure has another natural advantage: the volume of the boundary controls the typical distance from a random input to the boundary of its linear region (see §2.2). This measures the stability of the function computed by the network, and it is intuitively related to robustness under adversarial perturbation.



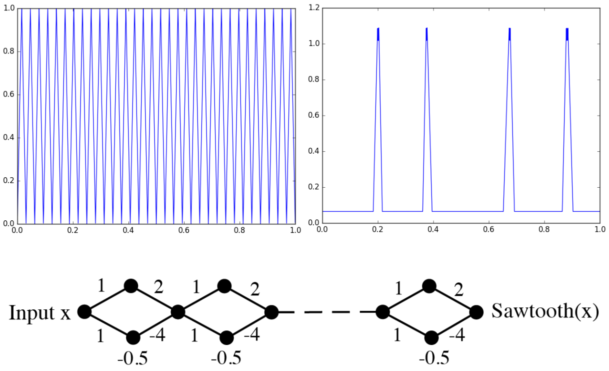


Figure 3. The sawtooth function on the left with 2^n teeth can be expressed succinctly by a ReLU network with only $3n + 4$ neurons (construction from Telgarsky (2015)). However, slight perturbation of the weights and biases of the network (by Gaussian noise with standard deviation 0.1) greatly simplifies the linear regions captured by the network.

Our Contributions. In this paper, we provide mathematical tools for analyzing the complexity of linear regions of a network with piecewise linear activations (such as ReLU) before, during, and after training. Our main contributions are as follows:

- For networks at initialization, the total surface area of the boundary between linear regions scales as the number of neurons times the number of breakpoints of the activation function. This is our main result, from which several corollaries follow (see Theorem 3, Corollary 4, and the discussion in §2).
- In particular, for any line segment through input space, the average number of regions intersecting it is *linear in the number of neurons*, far below the exponential number of regions that is theoretically attainable.
- Theorem 3 also allows us to conclude that, at initialization, the average distance from a sample point to the nearest region boundary is bounded below by a constant times the reciprocal of the number of neurons (see Corollary 5).
- We find empirically that both the number of regions and the distance to the nearest region boundary stay roughly constant during training and in particular are far from their theoretical maxima. That this should be the case is strongly suggested by Theorem 3, though not a direct consequence of it.

Overall, our results stress that practical expressivity lags significantly behind theoretical expressivity. Moreover, both our theoretical and empirical findings suggest that for certain measures of complexity, trained deep networks are remarkably similar to the same networks at initialization.

In the next section, we informally state our theoretical and empirical results and explore the underlying intuitions. Detailed descriptions of our experiments are provided in §3. The precise theorem statements for ReLU networks can be found in §5. The exact formulations for general piecewise linear networks are in Appendix A, with proofs in the rest of the Supplementary Material. In particular, Appendix B contains intuition for how our proofs are shaped, while details are completed in §C-D.

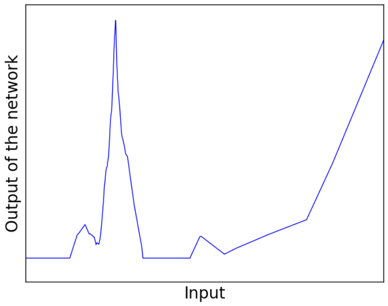


Figure 4. Graph of function computed by a ReLU net with input and output dimension 1 at initialization. The weights of the network are He normal (i.i.d. normal with variance = $2/\text{fan-in}$) and the biases are i.i.d. normal with variance 10^{-6}.

2. Informal Overview of Results

This section gives an informal introduction to our results. We begin in §2.1 by describing the case of networks with input dimension 1. In §2.2, we consider networks with higher input dimension. For simplicity, we focus throughout this section on fully connected ReLU networks. We emphasize, however, that our results apply to any piecewise linear activation. Moreover, the upper bounds we present in Theorems 1, 2, and 3 (and hence in Corollaries 4 and 5) can also be generalized to hold for feed-forward networks with arbitrary connectivity, though we do not go into details in this work, for the sake of clarity of exposition.

2.1. Number of Regions in 1D

Consider the simple case of a ReLU net $\mathcal{N}$ with input and output dimensions equal to 1. Such a network computes a piecewise linear function (see Figure 4), and we are interested in understanding both at initialization and during training the number of distinct linear regions. There is a simple universal upper bound:

Each activation can only
seprate the previous space in two parts.

$$\max \#\{\text{regions}\} \;\leq\; 2^{\#\text{neurons}}, \qquad (1)$$

where the maximum is over all settings of weight and biases. This bound depends on the architecture of $\mathcal{N}$ only via the number of neurons. For more refined upper bounds which take into account the widths of the layers, see Theorem 1 in Raghu et al. (2017) and Theorem 1 in Serra et al. (2018).

The constructions in Montufar et al. (2014); Telgarsky (2015); Raghu et al. (2017); Serra et al. (2018) indicate that the bound in (1) is very far from sharp for shallow and wide networks but that exponential growth in the number of regions can be achieved in deep, skinny networks for very special choices of weights and biases. This is a manifestation of the expressive power of depth, the idea that repeated compositions allow deep networks to capture complex hierarchical relations more efficiently per parameter than their shallow cousins. However, there is no non-trivial lower bound for the number of linear regions:

$$\min \#\{\text{regions}\} \;=\; 1, \qquad \forall \mathcal{N}.$$

The minimum is attained by setting all weights and biases to 0. This raises the question of the behavior for the average number of regions when the weights and biases are chosen at random (e.g. at initialization). Intuitively, configurations of weights and biases that come close to saturating the exponential upper bound (1) are numerically unstable in the sense that a small random perturbation of the weights and biases drastically reduces the number of linear regions (see Figure 3 for an illustration). In this direction, we prove a somewhat surprising answer to the question of how many regions $\mathcal{N}$ has at initialization. We state the result for ReLU but note that it holds for any piecewise linear, continuous activation function (see Theorems 3 and 6).

Because then the output space
is just {0}, so it has one region

Theorem 1 (informal)**.** *Let $\mathcal{N}$ be a network with piecewise linear activation with input and output dimensions of $\mathcal{N}$ both equal 1. Suppose the weights and biases are randomly initialized so that for each neuron z, its pre-activation $z(x)$ has bounded mean gradient*

$$\mathbb{E}\left[\|\nabla z(x)\|\right] \leq C, \qquad \textit{some } C > 0. \qquad (2)$$

This holds, for example, for ReLU *networks initialized with independent, zero-centered weights with variance $2/\text{fan-in}$. Then, for each subset $I \subset \mathbb{R}$ of inputs, the average number of linear regions inside I is proportional to the number of neurons times the length of I*

$$\mathbb{E}\left[\#\{\text{regions in } I\}\right] \;\approx\; |I| \cdot T \cdot \#\{\text{neurons}\},$$

where T is the number of breakpoints in the non-linearity of $\mathcal{N}$ (for ReLU nets, $T = 1$). The same result holds when computing the number of linear regions along any fixed 1-dimensional curve in a high-dimensional input space.

This theorem implies that the average number of regions along a one-dimensional curve in input space is proportional to the number of neurons, but independent of the arrangement of those neurons. In particular, a shallow network and a deep network will have the same complexity, by this measure, as long as they have the same total number of neurons. Of course, as $|I|$ grows, the bounds in Theorem 1 become less sharp. We plan to extend our results to obtain bounds on the total number of regions on all of $\mathbb{R}$ in the future. In particular, we believe that at initialization the mean total number of linear regions $\mathcal{N}$ is proportional to the number of neurons (this is borne out in Figure 5, which computes the total number of regions on an infinite line).

Theorem 1 defies the common intuition that, on average, each layer in $\mathcal{N}$ multiplies the number of regions formed up to the previous layer by a constant larger than one. This would imply that the average number of regions is exponential in the depth. To provide intuition for why this is not true for *random* weights and biases, consider the effect of each neuron separately. Suppose the pre-activation $z(x)$ of a neuron z satisfies $|z'(x)| = \Theta(1)$, a hallmark of any reasonable initialization. Then, over a compact set of inputs, the piecewise linear function $x \mapsto z(x)$ cannot be highly oscillatory over a large portion of the range of z. Thus, if the bias b_z is not too concentrated on any interval, we expect the equation $z(x) = b_z$ to have $O(1)$ solutions. On average, then, we expect that each neuron adds a *constant number* of new linear regions. Thus, the average total number of regions should scale roughly as the number of neurons.

Theorem 1 follows from a general result, Theorem 3, that holds for essentially any non-degenerate distribution of weights and biases and with any input dimension. If $\|\nabla z(x)\|$ and the bias distribution ρ_{b_z} are well-behaved, then throughout training, Theorem 3 suggests the number of linear regions along a 1-dimensional curve in input space scales like the number of neurons in $\mathcal{N}$. Figures 5-6 show experiments that give empirical verification of this heuristic.

2.2. Higher-Dimensional Regions

For networks with input dimension exceeding 1, there are several ways to generalize counting linear regions. A unit-matching heuristic applied to Theorem 1 suggests

$$\#\{\text{regions}\} \;=\; \#\{\text{neurons}\}^{n_{\text{in}}}, \qquad n_{\text{in}} = \text{input dim.}$$

Proving this statement is work in progress by the authors. Instead, we consider here a natural and, in our view, equally important generalization. Namely, for a bounded $K \subset \mathbb{R}^{n_{\text{in}}}$, we consider the $(n_{\text{in}} - 1)$-dimensional volume density

$$\mathrm{vol}_{n_{\text{in}}-1}\left(\mathcal{B}_{\mathcal{N}} \cap K\right) \big/ \mathrm{vol}_{n_{\text{in}}}(K), \tag{3}$$

where

$$\mathcal{B}_{\mathcal{N}} = \{x \mid \nabla\mathcal{N}(x) \text{ is not continuous at } x\} \tag{4}$$

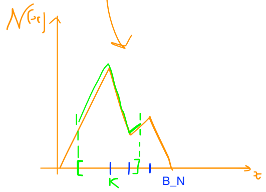
is the boundary of the linear regions for $\mathcal{N}$. When $n_{\text{in}} = 1$,

$$\mathrm{vol}_0\left(\mathcal{B}_{\mathcal{N}} \cap K\right) \;+\; 1 \;\;=\;\; \#\{\text{regions in } K\},$$

and hence the volume density (3) truly generalizes to higher input dimension of the number of regions. One reason for studying the volume density (3) is that it gives bounds from below for distance $(x, \mathcal{B}_{\mathcal{N}})$, which in turn provides insight into the nature of the computation performed by $\mathcal{N}$. Indeed, the exact formula

$$\text{distance}\left(x, \mathcal{B}_{\mathcal{N}}\right) = \min_{\text{neurons } z} \left\{|z(x) - b_z| \,/\, \|\nabla z(x)\|\right\},$$

shows that distance $(x, \mathcal{B}_{\mathcal{N}})$ measures the sensitivity over neurons at a given input x. In this formula, $z(x)$ denotes the pre-activation for a neuron z and b_z is its bias, so that $\mathrm{ReLU}(z(x) - b_z)$ is the post-activation. Moreover, the distance from a typical point to $\mathcal{B}_{\mathcal{N}}$ gives a heuristic lower bound for the typical distance to an adversarial example: two inputs closer than the typical distance to a linear region boundary likely fall into the same linear region, and hence are unlikely to be classified differently. Our next result generalizes Theorem 1.



Theorem 2 (informal)**.** *Let $\mathcal{N}$ be a network with a piecewise linear activation, input dimension n_{in} and output dimension* 1. *Suppose its weights and biases are randomly initialized as in* (2). *Then, for $K \subset \mathbb{R}^{d_{in}}$ bounded, the average volume of the linear region boundaries in K satisfies:*

$$\mathbb{E}\left[\frac{\mathrm{vol}_{n_{\text{in}}-1}\left(\mathcal{B}_{\mathcal{N}} \cap K\right)}{\mathrm{vol}_{n_{\text{in}}}(K)}\right] \;\approx\; T \cdot \#\{\text{neurons}\},$$

So counting
the number of linear regions
is like counting the number of neurons?
So why would you this measure instead
of the number of neurons?

This is only at initialization. Over training
this measure can give us insights about the
practical expressivness of the network. Which the
number of neurons cannot.

where T is the number of breakpoints in the non-linearity of $\mathcal{N}$ (for ReLU nets, $T = 1$). Moreover, if $x \in [0,1]^{n_{\text{in}}}$ is uniformly distributed, then the average, over both x and the weights/biases of $\mathcal{N}$, distance from x to $\mathcal{B}_{\mathcal{N}}$ satisfies

$$\mathbb{E}\left[\text{distance}\left(x, \mathcal{B}_{\mathcal{N}}\right)\right] \;\geq\; C\left(\#\{\text{neurons}\}\right)^{-1}, \quad C > 0.$$

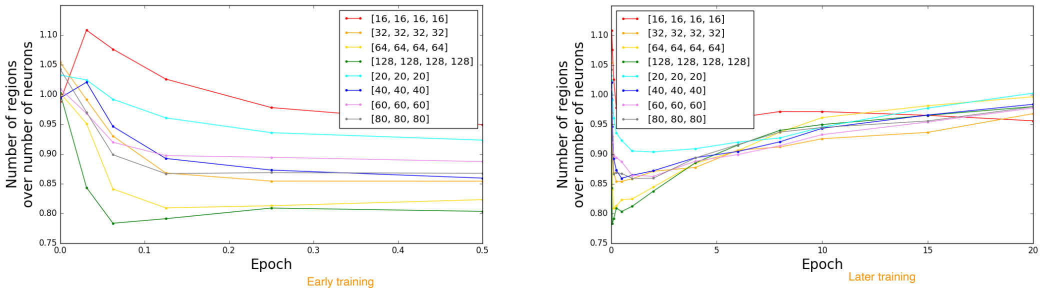
Experimentally, distance $(x, \mathcal{B}_{\mathcal{N}})$ remains comparable to $\left(\#\{\text{neurons}\}\right)^{-1}$ throughout training (see Figure 6).

3. Experiments

We empirically verified our theorems and further examined how linear regions of a network change during training. All experiments below were performed with fully-connected networks, initialized with He normal weights (i.i.d. with variance 2/fan-in) and biases drawn i.i.d. normal with variance 10^{-6} (to prevent collapse of regions at initialization, which occurs when all biases are uniquely zero). Training was performed on the vectorized MNIST (input dimension 784) using the Adam optimizer at learning rate 10^{-3}. All networks attain test accuracy in the range $95 - 98\%$.

3.1. Number of Regions Along a Line

We calculated the number of regions along lines through the origin and and a random selected training example in input space. For each setting of weights and biases within the network during training, the number of regions along each

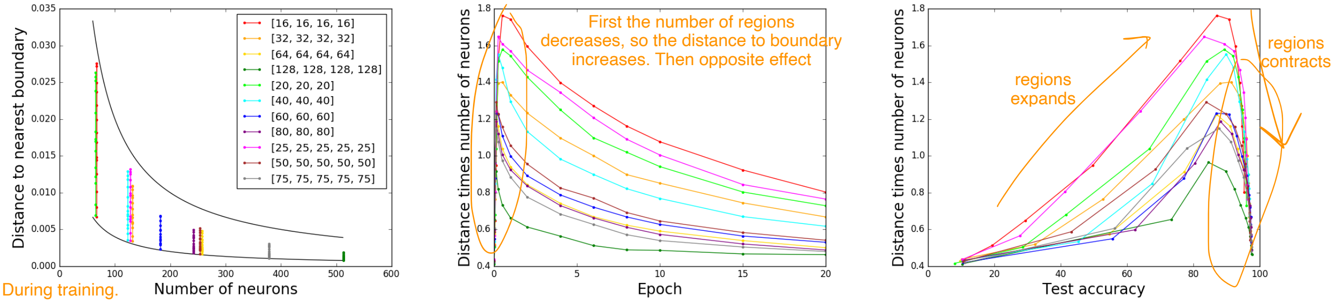


Early training

Later training

Lines are more or less flat around 1. So it looks like the number of regions crossing a 1D line during training doesn't grow during training

Figure 5. We here show how the number of regions along 1D lines in input space changes during training. In accordance with Theorem 3, we scale the number of regions by the number of neurons. Plots show (a) early training, up through 0.5 epochs, and (b) later training, up through 20 epochs. Note that for all networks, number of regions is a fixed constant times the number of neurons at initialization, as predicted, and that the number decreases (slightly) early in training before rebounding. $[n_1, n_2, n_3]$ in the legend corresponds to an architecture with layer widths 784 (input), n_1, n_2, n_3, 10 (output).



During training. One point for each epoch i guess

Figure 6. We here consider the average distance to the nearest boundary, as evaluated over 10000 randomly selected sample points. In (a) we show that this distance is essentially bounded between $0.4/\#\{\text{neurons}\}$ and $1.5/\#\{\text{neurons}\}$. Accordingly, in the next plot, we normalize the distance to the nearest boundary by dividing by the number of neurons. We plot this quantity against (b) epoch and (c) test accuracy. Observe that, in keeping with the findings of Figure 5, the distance to the nearest boundary first increases quickly (as the number of regions decreases), then rebounds more slowly as the network completes training. $[n_1, n_2, n_3]$ in the legend corresponds to an architecture with layer widths 784 (input), n_1, n_2, n_3, 10 (output).

line is calculated exactly by building up the network one layer at a time and calculating how each region is split by the next layer of neurons. Figure 5 represents the average over 5 independent runs, from each of which we sample 100 lines; variance across the different runs is not significant.

Figure 5 plots the average number of regions along a line, divided by the number of neurons in the network, as a function of epoch during training. We make several observations:

1. As predicted by Theorem 3, all networks start out with the number of regions along a line equal to a constant times the number of neurons in the network (the constant in fact appears very close to 1 in this case).
2. Throughout training, the number of regions does not deviate significantly from the number of neurons in the network, staying within a small constant of the value at initialization, in keeping with our intuitive understanding of Theorem 3 described informally around Theorem 1 above.
3. The number of regions actually decreases during the initial part of training, then increases again. We explore this behavior further in other experiments below.

So it looks like counting linear regions along a 1D line is not a good measure. It's more complex to compute than the number of neurons but stays more or less equal to it.
I guess this one is more to remark that the pratical epressiveness of NN is far from its theoretical maximum.

3.2. Distance to the Nearest Region Boundary

We calculated the average distance to the nearest boundary for 10000 randomly selected input points, for various networks throughout training. Points were selected randomly from a normal distribution with mean and variance matching the componentwise mean and variance of MNIST training data. Results were averaged over 12 independent runs, but variance across runs is not significant. Rerunning these experiments with sample points selected randomly from (i) the training data or (ii) the test data yielded similar results to random sample points.

In keeping with our results in the preceding experiment, the distance to the nearest boundary first increases then decreases during training. As predicted by Theorem 2, we find that for all networks, the distance to the nearest boundary is well-predicted by $1/\#\{\text{neurons}\}$. Throughout training, we find that it approximately varies between the curves $0.4/\#\{\text{neurons}\}$ and $1.5/\#\{\text{neurons}\}$ (Figure 6(a)). At initialization, as we predict, all networks have the same value for the product of number of neurons and distance to the nearest region boundary (Figure 6(b)); these products then diverge (slightly) for different architectures, first increasing rapidly and then decreasing more slowly.

We find Figure 6(c) fascinating, though we do not completely understand it. It plots the product of number of neurons and distance to the nearest region boundary against the test accuracy. It suggests two phases of training: first regions expand, then they contract. This lines up with observations made in Arpit et al. (2017) that neural networks "learn patterns first" on which generalization is simple and then refine the fit to encompass memorization of individual samples. A generalization phase would suggest that regions are growing, while memorization would suggest smaller regions are fit to individual data points. This is, however, speculation and more experimental (and theoretical) exploration will be required to confirm or disprove this intuition. We found it instructive to consider the full distribution of distances from sample points to their nearest boundaries, rather than just the average. For a single network (depth 4, width 16), Figure 7 indicates that this distribution does not significantly change during training, although there appears to be a slight skew towards larger regions, in agreement with the findings in Novak et al. (2018). The histogram shows log-distances. Hence, distance to the nearest region boundary varies over many orders of magnitude. This is consistent with Figures 1 and 4, which lend credence to the intuition that small distances to the nearest region boundary are explained by the presence of many small regions. According to Theorem 3, this should correlate with a combination of regions in input space at which some neurons have a large gradient and neurons with highly peaked biases distributions. We hope to return to this in future work.

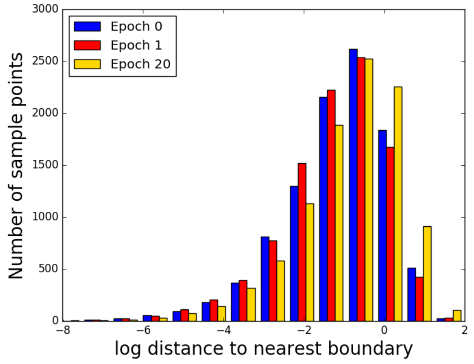


Figure 7. Distribution of log distances from random sample points to the nearest region boundary for a network of depth 4 and width 16, at initialization and after 1 and 20 epochs of training on MNIST.

3.3. Regions Within a 2D Plane

We visualized the regions of a network through training. Specifically, following experiments in Novak et al. (2018), we plotted regions within a plane in the 784-dimensional input space (Figure 8) through three data points with different labels (0, 1, and 2, in our case) inside a square centered at the circumcenter of the three examples. The network shown has depth 3 and width 64. We observe that, as expected from our other plots, the regions expand initially during training and then contract again. We expect the number of regions within a 2-dimensional subspace to be on the order of the square of the number of neurons – that is, $(64{\cdot}3)^2 \approx 4\times10^4$, which we indeed find.

Our approach for calculating regions is simple. We start with a single region (in this case, the square), and subdivide it by adding neurons to the network one by one. For each new neuron, we calculate the linear function it defines on each region, and determine whether that region is split into two. This approach terminates within a reasonable amount of time precisely because our theorem holds: there are relatively few regions. Note that we *exactly* determine all regions within the given square by calculating all region boundaries; thus our counts are exact and do not miss any small regions, as might occur if we merely estimated regions by sampling points from input space.

4. Related Work

There are a number of works that touch on the themes of this article: (i) the expressivity of depth; (ii) counting the number of regions in networks with piecewise linear activations; (iii) the behavior of linear regions through training; and (iv) the difference between expressivity and learnability. Related to (i), we refer the reader to Eldan & Shamir (2016); Telgarsky (2016) for examples of functions that can be efficiently represented by deep but not shallow ReLU nets. Next, still related to (i), for uniform approximation over classes of functions, again using deep ReLU nets, see Yarotsky (2017); Rolnick & Tegmark (2018); Yarotsky (2018); Petersen & Voigtlaender (2018). For interesting results on (ii) about counting the maximal possible number of linear regions in networks with piecewise linear activations see Bianchini & Scarselli (2014); Montufar et al. (2014); Poole et al. (2016); Arora et al. (2018); Raghu et al. (2017). Next, in the vein of (iii), for both a theoretical and empirical perspective on the number of regions computed by deep networks and specifically how the regions change during training, see Poole et al. (2016); Novak et al. (2018). In the direction of (iv), we refer the reader to Shalev-Shwartz et al. (2018); Hanin & Rolnick (2018); Hanin (2018). Finally, for general insights into learnability and expressivity in deep vs. shallow networks see Mhaskar & Poggio (2016); Mhaskar et al. (2016); Zhang et al. (2017); Lin et al. (2017); Poggio et al.

Paper suggests that number of linear regions scale like number of neurons.
Overall it is more or less the same at initizialization and after training.
But it seems to track different stages of training: 1. Expansion phase 2. Contraction phase
The distance ot the boundary seems to measure the same phenomenon with the 2 stages.
So it can be useful to choose the easiest one and use it to track what's happening
during training.

Less regions, but larger

More regions, but smaller

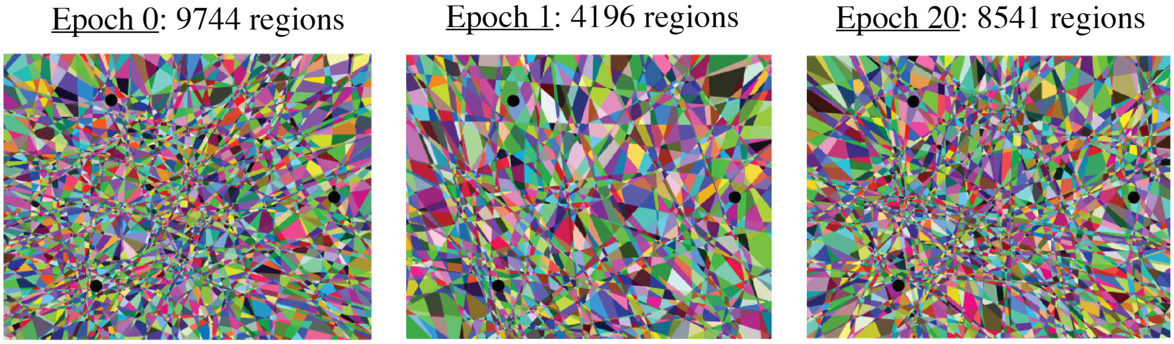


Figure 8. Here we show the linear regions that intersect a 2D plane through input space for a network of depth 3 and width 64 trained on MNIST. Black dots indicate the positions of the three MNIST training examples defining the plane. Note that we obtain qualitatively different pictures from Novak et al. (2018), which may result partially from our using ReLU activation instead of ReLU6.

(2017); Neyshabur et al. (2017).

5. Formal Statement of Results

To state our results precisely, we fix some notation. Let $d, n_{\text{in}}, n_1, \dots, n_d \geq 1$ and consider a depth d fully connected ReLU net $\mathcal{N}$ with input dimension n_{in}, output dimension 1, and hidden layer widths n_j, $j = 1, \dots, d-1$. As explained in the introduction, a generic configuration of its weights and biases partitions the input space $\mathbb{R}^{n_{\text{in}}}$ into a union of polytopes P_j with disjoint interiors. Restricted to each P_j, $\mathcal{N}$ computes a linear function.

Our main mathematical result, Theorem 3, concerns the set $\mathcal{B}_{\mathcal{N}}$ of points $x \in \mathbb{R}^{n_{\text{in}}}$ at which the gradient $\nabla\mathcal{N}$ is discontinuous at x (see (4)). For each $k = 1, \dots, n_{\text{in}}$, we define

$$\mathcal{B}_{\mathcal{N},k} = \text{the “}(n_{\text{in}} - k)\text{–dimensional piece” of } \mathcal{B}_{\mathcal{N}}. \tag{5}$$

More precisely, we set $\mathcal{B}_{\mathcal{N},0} := \emptyset$ and recursively define $\mathcal{B}_{\mathcal{N},k}$ to be the set of points $x \in \mathcal{B}_{\mathcal{N}} \backslash \{\mathcal{B}_{\mathcal{N},0} \cup \cdots \cup \mathcal{B}_{\mathcal{N},k-1}\}$ so that in a neighborhood of x the set $\mathcal{B}_{\mathcal{N}} \backslash \{\mathcal{B}_{\mathcal{N},0} \cup \cdots \cup \mathcal{B}_{\mathcal{N},k-1}\}$ coincides with a co-dimension k hyperplane.

For example, when $n_{\text{in}} = 2$, the linear regions P_j are polygons, the set $\mathcal{B}_{\mathcal{N},1}$ is the union of the open line segments making up the boundaries of the P_j, and $\mathcal{B}_{\mathcal{N},2}$ is the collection of vertices of the P_j. Theorem 3 provides a convenient formula for the average of the $(n_{\text{in}} - k)-$dimensional volume of $\mathcal{B}_{\mathcal{N},k}$ inside any bounded, measurable set $K \subset \mathbb{R}^{n_{\text{in}}}$. To state the result, for every neuron z in $\mathcal{N}$ we will write

$$z(x) \;:=\; \text{pre-activation at } z, \quad \ell(z) \;=\; \text{layer index of } z \tag{6}$$

and $b_z \;:=\;$ bias at z. Thus, for a given input $x \in \mathbb{R}^{n_0}$, the post-activation of z is

$$Z(x) := \text{ReLU}(z(x)) = \max\{0, z(x) - b_z\}. \tag{7}$$

Theorem 3 holds under the following assumption on the distribution of weights and biases:

A1: The conditional distribution of any collection of biases $b_{z_1}, \dots, b_{z_k}$, given all the other weights and biases, has a density $\rho_{b_{z_1},\dots,b_{z_k}}(b_1, \dots, b_k)$ with respect to Lebesgue measure on $\mathbb{R}^k$.

A2: The joint distribution of all the weights has a density with respect to Lebesgue measure on $\mathbb{R}^{\#\text{weights}}$.

These assumptions hold in particular when the weights and biases of $\mathcal{N}$ are independent with marginal distributions that have a density relative to Lebesgue measure on $\mathbb{R}$ (i.e. at initialization). They hold much more generally, however, and can intuitively be viewed as a non-degeneracy assumption on the behavior of the weights and biases of $\mathcal{N}$. Specifically, they are used in Proposition 10 to ensure that the set $\mathcal{B}_{\mathcal{N},k}$ consists of inputs where exactly k neurons turn off/on. Assumption (A1) also allows us, in Proposition 11, to apply the co-area formula (29) to compute the expect volume of the set of inputs where a given collection of neurons turn on/off. Our main result is the following.

Theorem 3. *Suppose $\mathcal{N}$ is a feed-forward* ReLU *net with input dimension n_0, output dimension 1, and random weights/biases. Assume that the distribution of weights/biases satisfies Assumptions A1 and A2 above. Then, with the notation* (6)*, for any bounded measurable set $K \subseteq \mathbb{R}^{n_{\text{in}}}$ and any $k = 1, \dots, n_{\text{in}}$, the average $(n_{\text{in}} - k)-$dimensional volume $\mathbb{E}\left[\text{vol}_{n_{\text{in}}-k}(\mathcal{B}_{\mathcal{N},k} \cap K)\right]$ of $\mathcal{B}_{\mathcal{N},k}$ inside K is*

$$\mathbb{E}\left[\text{vol}_{n_{\text{in}}-k}(\mathcal{B}_{\mathcal{N},k} \cap K)\right] \tag{8}$$

of $\mathcal{B}_{\mathcal{N},k}$ inside K is, in the notation (6)*,*

$$\sum_{\substack{\text{distinct neurons} \\ z_1,\dots,z_k \text{ in } \mathcal{N}}} \int_K \mathbb{E}\left[Y_{z_1,\dots,z_k}(x)\right] dx,$$

where $Y_{z_1,\dots,z_k}(x)$ is

$$\left\|J_{z_1,\dots,z_k}(x)\right\| \, \rho_{b_{z_1},\dots,b_{z_k}}(z_1(x), \dots, z_k(x))$$

times the indicator function of the event that z_j is good at x for each $j = 1, \ldots, k$. Here, $J_{z_1,\ldots,z_k}$ is the $k \times n_{\text{in}}$ Jacobian of the map $x \mapsto (z_1(x), \ldots, z_k(x))$,

$$\|J_{z_1,\ldots,z_k}(x)\| := \det\left(J_{z_1,\ldots,z_k}(x)\,(J_{z_1,\ldots,z_k}(x))^T\right)^{1/2},$$

the function $\rho_{b_{z_1},\ldots,b_{z_k}}$ is the density of the joint distribution of the biases $b_{z_1}, \ldots, b_{z_k}$, and we say a neuron z is good at x if there exists a path of neurons from z to the output in the computational graph of $\mathcal{N}$ so that each neuron along this path is open at x).

To evaluate the expression in (8) requires information on the distribution of gradients $\nabla z(x)$, the pre-activations $z(x)$, and the biases b_z. Exact information about these quantities is available at initialization (Hanin, 2018; Hanin & Rolnick, 2018; Hanin & Nica, 2018), yielding the following Corollary.

Corollary 4. *With the notation and assumptions of Theorem 3, suppose the weights are independent are drawn from a fixed probability measure μ on $\mathbb{R}$ that is symmetric around* 0 *and then rescaled to have* $\mathrm{Var}[\text{weights}] = 2/$*fan-in. Fix $k \in \{1, \ldots, n_{\text{in}}\}$. Then there exists $C > 0$ for which*

$$\begin{aligned}\frac{\mathbb{E}\left[\mathrm{vol}_{n_{\text{in}}-k}(\mathcal{B}_{\mathcal{N},k} \cap K)\right]}{\mathrm{vol}_{n_{\text{in}}}(K)} & \\ \leq \binom{\#\{\text{neurons}\}}{k} (C_{\text{grad}} \cdot C_{\text{bias}})^k,\end{aligned} \tag{9}$$

where

$$C_{\text{bias}} = \sup_z \sup_{b \in \mathbb{R}} \rho_{b_z}(b)$$

and

$$C_{\text{grad}} = \sup_z \sup_{x \in \mathbb{R}^{n_{\text{in}}}} \mathbb{E}\left[\|\nabla z(x)\|^{2k}\right]^{1/k} \leq Ce^{C \sum_{j=1}^d \frac{1}{n_j}}$$

where $C > 0$ depends only on μ but not on the architecture of $\mathcal{N}$ and n_j is the width of the j^{th} hidden layer. Moreover, we also have similar lower bounds

$$\binom{\#\{\text{neurons}\}}{k} c_{\text{bias}}^k \leq \frac{\mathbb{E}\left[\mathrm{vol}_{n_{\text{in}}-k}(\mathcal{B}_{\mathcal{N},k} \cap K)\right]}{\mathrm{vol}_{n_{\text{in}}}(K)} \tag{10}$$

where

$$c_{\text{bias}} = \inf_{|b| \leq \eta} \rho_{b_z}(b),$$

and

$$\eta = \left(\frac{\sup_{x \in K} \|x\|^2}{n_{\text{in}}} + \sum_{j=1}^d \sigma_{b_j}^2\right) e^{C' \sum_{j=1}^d \frac{1}{n_j}},$$

with $C' > 0$ depending only on the distribution μ of the weights in $\mathcal{N}$.

We prove Corollary 7 in Appendix D. Let us state one final corollary of Theorem 3

Corollary 5. *Suppose $\mathcal{N}$ is as in Theorem 3 and satisfies the hypothesis* (14) *in Corollary 7. Then, for any compact set $K \subset \mathbb{R}^{n_{\text{in}}}$ let x be a uniform point in K. There exists $c > 0$ independent of K so that*

$$\mathbb{E}\left[\text{distance}(\text{x}, \mathcal{B}_{\mathcal{N}})\right] \geq \frac{c}{C_{\text{bias}} C_{\text{grad}} \#\{\text{neurons}\}}.$$

We prove Corollary 8 in §E. The basic idea is simple. For every $\epsilon > 0$, we have

$$\mathbb{E}\left[\text{distance}(\text{x}, \mathcal{B}_{\mathcal{N}})\right] \geq \epsilon \mathbb{P}\left(\text{distance}(x, \mathcal{B}_{\mathcal{N}}) > \epsilon\right),$$

with the probability on the right hand side scaling like

$$1 - \mathrm{vol}_{n_{\text{in}}}(T_\epsilon(\mathcal{B}_{\mathcal{N}}) \cap K) / \mathrm{vol}_{n_{\text{in}}}(K),$$

where $T_\epsilon(\mathcal{B}_{\mathcal{N}})$ is the tube of radius ϵ around $\mathcal{B}_{\mathcal{N}}$. We expect that its volume like $\epsilon \,\mathrm{vol}_{n_{\text{in}}-1}(\mathcal{B}_{\mathcal{N}})$. Taking $\varepsilon = c/\#\{\text{neurons}\}$ yields the conclusion of Corollary 8.

6. Conclusions and Further Work

The question of why depth is powerful has been a persistent problem for deep learning theory, and one that recently has been answered by works giving enhanced expressivity as the ultimate explanation. However, our results suggest that such explanations may be misleading. While we do not speak to all notions of expressivity in this paper, we have both theoretically and empirically evaluated one common measure: the linear regions in the partition of input space defined by a network with piecewise linear activations. We found that the average size of the boundary of these linear regions depends only on the number of neurons and not on the network depth – both at initialization and during training. This strongly suggests that deeper networks do not learn more complex functions than shallow networks. We plan to test this interpretation further in future work – for example, with experiments on more complex tasks, as well as by investigating higher order statistics, such as the variance.

We do not propose a replacement theory for the success of deep learning; however, prior work has already hinted at how such a theory might proceed. Notably, Ba & Caruana (2014) show that, once deep networks are trained to perform a task successfully, their behavior can often be replicated by shallow networks, suggesting that the advantages of depth may be linked to easier learning.

References

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. In *ICLR*, 2018.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *ICML*, 2017.

Ba, J. and Caruana, R. Do deep nets really need to be deep? In *NeurIPS*, pp. 2654–2662, 2014.

Bianchini, M. and Scarselli, F. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, 2014.

Eldan, R. and Shamir, O. The power of depth for feedforward neural networks. In *COLT*, pp. 907–940, 2016.

Hanin, B. Which neural net architectures give rise to exploding and vanishing gradients? In *NeurIPS*, 2018.

Hanin, B. and Nica, M. Products of many large random matrices and gradients in deep neural networks. *Preprint arXiv:1812.05994*, 2018.

Hanin, B. and Rolnick, D. How to start training: The effect of initialization and architecture. In *NeurIPS*, 2018.

Lin, H. W., Tegmark, M., and Rolnick, D. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

Mhaskar, H., Liao, Q., and Poggio, T. Learning functions: when is deep better than shallow. *Preprint arXiv:1603.00988*, 2016.

Mhaskar, H. N. and Poggio, T. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *NeurIPS*, pp. 2924–2932, 2014.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *NeurIPS*, pp. 5947–5956, 2017.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. In *ICLR*, 2018.

Petersen, P. and Voigtlaender, F. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. 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.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *NeurIPS*, pp. 3360–3368, 2016.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Dickstein, J. S. On the expressive power of deep neural networks. In *ICML*, pp. 2847–2854, 2017.

Rolnick, D. and Tegmark, M. The power of deeper networks for expressing natural functions. In *ICLR*, 2018.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. Bounding and counting linear regions of deep neural networks. In *ICML*, 2018.

Shalev-Shwartz, S., Shamir, O., and Shammah, S. Failures of gradient-based deep learning. In *ICML*, 2018.

Telgarsky, M. Representation benefits of deep feedforward networks. *Preprint arXiv:1509.08101*, 2015.

Telgarsky, M. Benefits of depth in neural networks. In *COLT*, 2016.

Yarotsky, D. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

Yarotsky, D. Optimal approximation of continuous functions by very deep ReLU networks. In *COLT*, 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.