Bounding and Counting Linear Regions of Deep Neural Networks

Thiago Serra [* 1]   **Christian Tjandraatmadja** [* 1]   **Srikumar Ramalingam** [2]

Abstract

We investigate the complexity of deep neural networks (DNN) that represent piecewise linear (PWL) functions. In particular, we study the number of linear regions, i.e. pieces, that a PWL function represented by a DNN can attain, both theoretically and empirically. We present (i) tighter upper and lower bounds for the maximum number of linear regions on rectifier networks, which are exact for inputs of dimension one; (ii) a first upper bound for multi-layer maxout networks; and (iii) a first method to perform exact enumeration or counting of the number of regions by modeling the DNN with a mixed-integer linear formulation. These bounds come from leveraging the dimension of the space defining each linear region. The results also indicate that a deep rectifier network can only have more linear regions than every shallow counterpart with same number of neurons if that number exceeds the dimension of the input.
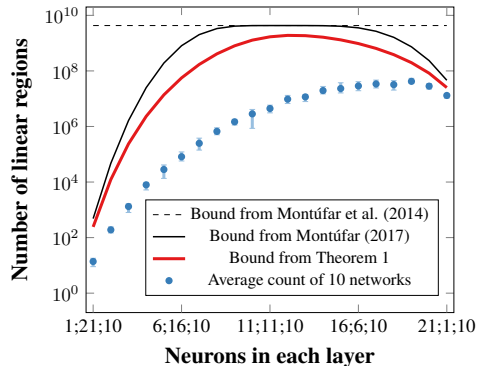
Figure 1. This paper shows improved bounds on the "number of linear regions" (typically used to study the expressiveness of DNNs) of PWL functions modeled by DNNs that use rectified linear activation functions, and a method for exact counting of the number of such regions in trained networks. We compare upper bounds from the first and latest results in the literature (Montúfar et al., 2014; Montúfar, 2017) with our main result (See Theorem 1). Using the proposed exact counting algorithm, we show the actual number of linear regions in 10 rectifier networks for MNIST digit recognition task with each configuration of two hidden layers totaling 22 neurons, reporting average and min-max range.

1. Introduction

We have seen an unprecedented success of DNNs in computer vision, speech, and other domains (Krizhevsky et al., 2012; Ciresan et al., 2012; Goodfellow et al., 2013; Hinton et al., 2012). While the popular networks such as AlexNet (Krizhevsky et al., 2012), GoogleNet (Szegedy et al., 2015), and residual networks (He et al., 2016) have shown record beating performance on various image recognition tasks, empirical results still govern the design of network architecture in terms of depth and activation functions. Two important considerations that are part of most successful architectures are greater depth and the use of PWL activation functions such as rectified linear units (ReLUs). This large gap between practice and theory has driven

researchers toward mathematical modeling of the expressive power of DNNs (Cybenko, 1989; Anthony & Bartlett, 1999; Pascanu et al., 2014; Montúfar et al., 2014; Bianchini & Scarselli, 2014; Eldan & Shamir, 2016; Telgarsky, 2015; Mhaskar et al., 2016; Raghu et al., 2017; Montúfar, 2017).

The expressiveness of DNNs can be studied by transforming one network to another with different number of layers or activation functions. While any continuous function can be modeled using a single hidden layer of sigmoid activation functions (Cybenko, 1989), shallow networks require exponentially more neurons to model functions that can be modeled using much smaller deeper networks (Delalleau & Bengio, 2011). There are a wide variety of activation functions, which come with different modeling capabilities, such as threshold ($f(z) = (z > 0)$), logistic ($f(z) = 1/(1 + \exp(-e))$), ReLU ($f(z) = \max\{0, z\}$), and maxout ($f(z_1, z_2, \ldots, z_k) = \max\{z_1, z_2, \ldots, z_k\}$). It has been shown that sigmoid networks are more expressive than similar-sized threshold networks (Maass et al., 1994), and ReLU networks are more expressive than similar-sized

---

[*]Equal contribution [1]Carnegie Mellon University, Pittsburgh, USA [2]The University of Utah, Salt Lake City, USA. Correspondence to: Thiago Serra <tserraaz@alumni.cmu.edu>, Christian Tjandraatmadja <ctjandra@alumni.cmu.edu>, Srikumar Ramalingam <srikumar@cs.utah.edu>.

threshold networks (Pan & Srikumar, 2016).

The complexity or expressiveness of neural networks belonging to the family of PWL functions can also be analyzed by looking at how the network can partition the input space to an exponential number of linear response regions (Pascanu et al., 2014; Montúfar et al., 2014). The basic idea of a PWL function is simple: we can divide the input space into several regions and we have individual linear functions for each of these regions. Functions partitioning the input space to a larger number of linear regions are considered to be more complex, or in other words, possess better representational power. In the case of ReLUs, it has been shown that some deep networks separate their input space into exponentially more linear response regions than their shallow counterparts despite using the same number of activation functions (Pascanu et al., 2014). The results were later extended and improved. In particular, Montúfar et al. (2014) show upper and lower bounds on the maximal number of linear regions for a ReLU DNN and a single layer maxout network, and a lower bound for a maxout DNN. Furthermore, Raghu et al. (2017) and Montúfar (2017) improve the upper bound for a ReLU DNN. This upper bound asymptotically matches the lower bound from Montúfar et al. (2014) when the number of layers and input dimension are constant and all layers have the same width. Finally, Arora et al. (2018) improve the lower bound by providing a family of ReLU DNNs with an exponential number of regions for fixed size and depth.

Main Contributions

This paper directly improves on the results of Montúfar et al. (Pascanu et al., 2014; Montúfar et al., 2014; Montúfar, 2017), Raghu et al. (2017), and Arora et al. (2018). Fig. 1 highlights the main contributions, and the following list summarizes all the contributions:

- We achieve tighter upper and lower bounds on the maximal number of linear regions of the PWL function corresponding to a DNN that employs ReLUs as shown in Fig. 1. As a special case, we present the exact maximal number of regions when the input dimension is one. We additionally provide the first upper bound for multi-layer maxout networks. (See Sections 3 and 4).

- We show for ReLUs that the exact maximal number of linear regions of shallow networks is larger than that of deep networks if the input dimension exceeds the number of neurons, a result that could not be inferred from bounds in prior work. (See Figure 5).

- We use a mixed-integer linear formulation to show that counting linear regions is indeed possible. For

the first time, we show the exact number of linear regions for a sample of DNNs as shown in Fig. 1. This new capability can be used to evaluate the tightness of the bounds and potentially to analyze the correlation between accuracy and the number of linear regions. (See Sections 5 and 6).

2. Notations and Background

Let us assume that a feedforward neural network, which is studied in this paper, has n_0 input variables given by $\mathbf{x} = \{x_1, x_2, \ldots, x_{n_0}\}$, and m output variables given by $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$. Each hidden layer $l = \{1, 2, \ldots, L\}$ has n_l hidden neurons whose activations are given by $\mathbf{h}^l = \{h_1^l, h_2^l, \ldots, h_{n_l}^l\}$. Let W^l be the $n_l \times n_{l-1}$ matrix where each row corresponds to the weights of a neuron of layer l. Let $\mathbf{b}^l$ be the bias vector used to obtain the activation functions of neurons in layer l. Based on the $\mathrm{ReLU}(x) = \max\{0, x\}$ activation function, the activations of the hidden neurons and the outputs are given below:

$$\begin{align} \mathbf{h}^1 &= \max\{0, W^1\mathbf{x} + b^1\} \\ \mathbf{h}^l &= \max\{0, W^l\mathbf{h}^{l-1} + b^l\} \\ \mathbf{y} &= W^{L+1}\mathbf{h}^{\mathbf{L}} \end{align}$$

As considered in Pascanu et al. (2014), the output layer is a linear layer that computes the linear combination of the activations from the previous layer without any ReLUs.

We can treat the DNN as a piecewise linear (PWL) function $F : \mathbb{R}^{n_0} \to \mathbb{R}^m$ that maps the input $\mathbf{x}$ in $\mathbb{R}^{n_0}$ to $\mathbf{y}$ in $\mathbb{R}^m$. We define linear regions based on activation patterns.

Activation Pattern: Let us consider an input vector $\mathbf{x} = \{x_1, \ldots, x_{n_0}\}$. For every layer l we define an activation set $S^l \subseteq \{1, \ldots, n_l\}$ such that $e \in S^l$ if and only if the ReLU e is active, i.e. $h_e^l > 0$. We aggregate these activation sets into $\mathcal{S} = (S^1, \ldots, S^l)$, which we call an activation pattern. In this work, we may consider activation patterns up to a layer $l \leq L$. Activation patterns were previously defined in terms of strings (Raghu et al., 2017; Montúfar, 2017). We say that an input $\mathbf{x}$ corresponds to an activation pattern $\mathcal{S}$ if feeding $\mathbf{x}$ to the DNN results in the activations in $\mathcal{S}$.

We define linear regions as follows.[1]

Definition 1. *Given a PWL function $F : \mathbb{R}^{n_0} \to \mathbb{R}^m$ represented by a DNN, a linear region is the set of inputs that correspond to a same activation pattern in the DNN.*

---

[1]There is a subtly different definition of linear region in the literature (Pascanu et al., 2014; Montúfar et al., 2014), as follows. Given a PWL function F, a linear region is a maximal connected subset of the input space on which F is linear. The two definitions are essentially the same, except in a few degenerate cases. There could be scenarios where two different activation patterns may correspond to two adjacent regions with the same linear function, in which case this definition considers them as a single one. However, the bounds derived in this paper are valid for both definitions.
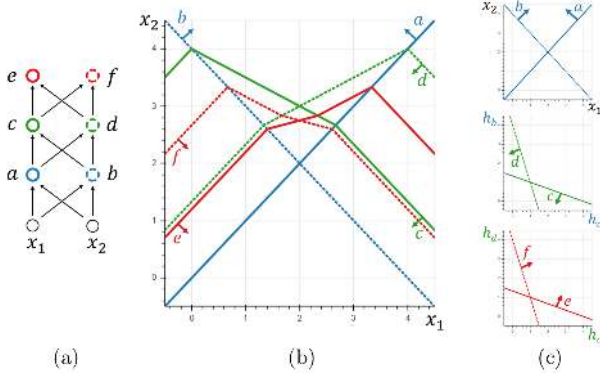
(a)   (b)   (c)

Figure 2. (a) Simple DNN with two inputs and three layers with 2 activation units each. (b) Visualization of the hyperplanes from the layers partitioning the input space into 20 linear regions. The arrows indicate the directions in which the corresponding neurons are activated. (c) Visualization of the hyperplanes from the first, second, and third (non-input) layers (from top to bottom) in the space given by the outputs of their respective previous layers. The outputs of the units are $h_a = \max\{0, -x_1 + x_2\}$, $h_b = \max\{0, x_1 + x_2 - 4\}$, $h_c = \max\{0, -h_a - 3h_b + 4\}$, $h_d = \max\{0, -3h_a - h_b + 4\}$, $h_e = \max\{0, h_c + 3h_d - 4\}$, and $h_f = \max\{0, 3h_c + h_d - 4\}$.

In this paper, we interchangeably refer to $\mathcal{S}$ as an activation pattern or a region for convenience.

In Figure 2 we show a simple ReLU DNN with two inputs $\{x_1, x_2\}$ and 3 layers. The activation units $\{a, b, c, d, e, f\}$ on these layers can be thought of as hyperplanes that each divide the space in two. On one side of the hyperplane, the unit outputs a positive value. For all points on the other side of the hyperplane including itself, the unit outputs 0.

One may wonder: into how many linear regions do n hyperplanes split a space? Zaslavsky (1975) shows that an arrangement of n hyperplanes divides a d-dimensional space into at most $\sum_{s=0}^{d} \binom{n}{s}$ regions, a bound that is attained when they are in general position. The term general position basically means that a small perturbation of the hyperplanes does not change the number of regions. This corresponds to the exact maximal number of regions of a single layer DNN with n ReLUs and input dimension d.

In Figures 2(b)–(c), we provide a visualization of how ReLUs partition the input space. Figure 2(c) shows the hyperplanes corresponding to the ReLUs at layers $l = 1, 2$, and 3, from top to bottom. Figure 2(b) considers these same hyperplanes in the input space x. If we consider only the first-layer hyperplanes, the 2D input space is partitioned into 4 regions, as per Zaslavsky (1975) $\left(\binom{2}{0} + \binom{2}{1} + \binom{2}{2} = 4\right)$. The regions are further partitioned as we consider additional layers. Subsequent hyperplanes are affected by the transformations applied in the earlier layers.
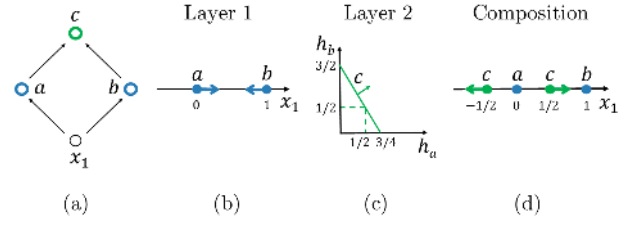

(a)   (b)   (c)   (d)

Figure 3. (a) A network with one input x_1 and three activation units a, b, and c. (b) We show the hyperplanes $x_1 = 0$ and $-x_1 + 1 = 0$ corresponding to the two activation units in the first hidden layer. In other words, the activation units are given by $h_a = \max\{0, x_1\}$ and $h_b = \max\{0, -x_1 + 1\}$. (c) The activation unit in the third layer is given by $h_c = \max\{0, 4h_a + 2h_b - 3\}$. (d) The activation boundary for neuron c is disconnected.

Figure 2 also highlights that activation boundaries behave like hyperplanes when inside a region and may bend whenever they intersect with a boundary from a previous layer. This has also been pointed out by Raghu et al. (2017). In particular, they cannot appear twice in the same region as they are defined by a single hyperplane if we fix the region. Moreover, these boundaries do not need to be connected, as illustrated in Figure 3.

3. Tighter Bounds for Rectifier Networks

Montúfar et al. (2014) derive an upper bound of 2^N for N units, which can be obtained by mapping linear regions to activation patterns. Raghu et al. (2017) improve this result by deriving an asymptotic upper bound of $O(n^{Ln_0})$ to the maximal number of regions, assuming $n_l = n$ for all layers l and $n_0 = O(1)$. Montúfar (2017) further tightens the upper bound to $\prod_{l=1}^{L} \sum_{j=0}^{d_l} \binom{n_l}{j}$, where $d_l = \min\{n_0, n_1, \ldots, n_l\}$.

Moreover, Montúfar et al. (2014) prove a lower bound of $\left(\prod_{l=1}^{L-1} \lfloor n_l/n_0 \rfloor^{n_0}\right) \sum_{j=0}^{n_0} \binom{n_L}{j}$ when $n \geq n_0$, or asymptotically $\Omega((n/n_0)^{(L-1)n_0} n^{n_0})$. Arora et al. (2018) present a lower bound of $2 \sum_{j=0}^{n_0-1} \binom{m-1}{j} w^{L-1}$ where $2m = n_1$ and $w = n_l$ for all $l = 2, \ldots, L$. We derive both upper and lower bounds that improve upon these previous results.

3.1. An Upper Bound on the Number of Linear Regions

In this section, we prove the following upper bound on the number of regions.

Theorem 1. *Consider a deep rectifier network with L layers, n_l rectified linear units at each layer l, and an input of dimension n_0. The maximal number of regions of this*

neural network is at most

$$\sum_{(j_1,\dots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l}$$

where $J = \{(j_1,\dots,j_L)\in\mathbb{Z}^L : 0 \leq j_l \leq \min\{n_0, n_1 - j_1,\dots,n_{l-1} - j_{l-1}, n_l\}\ \forall l = 1,\dots,L\}$. *This bound is tight when* $L = 1$.

Note that this is a stronger upper bound than the one that appeared in Montúfar (2017), which can be derived from this bound by relaxing the terms $n_l - j_l$ to n_l and factoring the expression. When $n_0 = O(1)$ and all layers have the same width n, we have the same best known asymptotic bound $O(n^{Ln_0})$ first presented in Raghu et al. (2017).

Two insights can be extracted from the above expression:

1. **Bottleneck effect.** The bound is sensitive to the positioning of layers that are small relative to the others, a property we call the bottleneck effect. If we subtract a neuron from one of two layers with the same width, choosing the one closer to the input layer will lead to a larger (or equal) decrease in the bound. This occurs because each index j_l is essentially limited by the widths of the current and previous layers, $n_0, n_1,\dots,n_l$. In other words, smaller widths in the first few layers of the network imply a bottleneck on the bound.

   The following proposition illustrates this bottleneck effect of the bound for the 2-layer case (see Appendix A for the proof).

   Proposition 2. *Consider a 2-layer network with widths* n_1, n_2 *and input dimension* $n_0 > \max\{n_1, n_2\}$. *Then moving a neuron from the first layer to the second layer strictly decreases the bound.*

   Figure 4 illustrates this behavior. For the solid line, we keep the total size of the network the same but shift from a small-to-large network (i.e., smaller width near the input layer and larger width near the output layer) to a large-to-small network in terms of width. We see that the bound monotonically increases as we reduce the bottleneck. If we add a layer of constant width at the end, represented by the dashed line, the bound decreases when the layers before the last become too small and create a bottleneck for the last layer.

   While this is a property of the upper bound rather than of the exact maximal number of regions, we observe in Section 6 that empirical results for the number of regions of a trained network exhibit a behavior that resembles the bound as the width of the layers vary. Moreover, this bottleneck effect appears at a more fundamental level. For example, having a first layer of size

one forces all hyperplanes corresponding to subsequent layers to be parallel to each other in the input space, reflecting the fact that we have compressed all information into a single value. More generally, the smaller a layer is, the more linearly dependent the hyperplanes from subsequent layers will be, which results in fewer regions. Further in this section, we formalize this in terms of dimension and show that the dimension of the image of a region is limited by the widths of earlier layers, which is used to prove Theorem 1.

2. **Deep vs shallow for large input dimensions.** In several applications such as imaging, the input dimension can be very large. Montúfar et al. (2014) show that if the input dimension n_0 is constant, then the number of regions of deep networks is asymptotically larger than that of shallow (single-layer) networks. We complement this picture by establishing that if the input dimension is large, then shallow networks can attain more regions than deep networks.

   More precisely, we compare a deep network with L layers of equal width n and a shallow network with one layer of width Ln. Denote the exact maximal number of regions by $R(n_0, n_1,\dots,n_L)$, where n_0 is the input dimension and $n_1,\dots,n_L$ are the widths of layers 1 through L of the network.

   Corollary 3. *Let* $L \geq 2$, $n \geq 1$, *and* $n_0 \geq Ln$. *Then*

   $$R(n_0, \underbrace{n,\dots,n}_{L\ times}) < R(n_0, Ln)$$

   Moreover, $\lim_{L\to\infty} \frac{R(n_0,n,\dots,n)}{R(n_0,Ln)} = 0$.

   This is a consequence of Theorem 1 (see Appendix A for the proof).

   Figure 5 (a) illustrates this behavior. As we increase the number of layers while keeping the total size of the network constant, the bound plateaus at a value lower than the exact maximal number of regions for shallow networks. Moreover, the number of layers that yields the highest bound decreases as we increase the input dimension n_0. Hence, for a given number of units and n_0, there is a particular depth maximizing the bound.

   This property cannot be inferred from upper bounds derived in prior work: previous bounds for a network with L layers of size n are no smaller than $R(n_0, Ln)$ for a sufficiently large n_0. Figure 5 (b) shows the behavior of the previous best known bound.

   We remark that asymptotically both deep and shallow networks can attain exponentially many regions when the input dimension is sufficiently large. More precisely, for a DNN with the same width n per layer, the maximal number of linear regions is $\Omega(2^{\frac{2}{3}Ln})$ when $n_0 \geq n/3$ (see Appendix B).
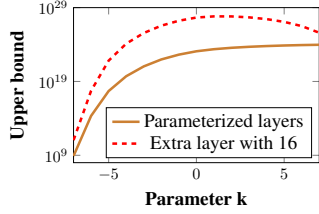
Figure 4. Upper bound from Theorem 1, in semilog scale, for input dimension $n_0 = 32$ and the width of the first five layers parameterized as $16 + 2k, 16 + k, 16, 16 - k, 16 - 2k$.
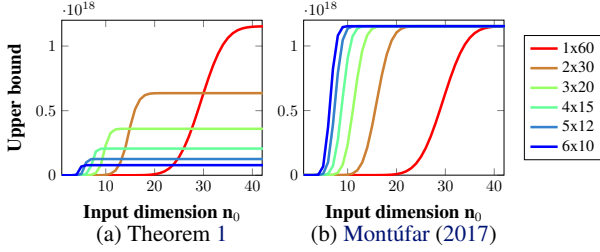

Figure 5. Comparison of bounds for evenly distributing 60 neurons in 1 to 6 layers, where the single-layer case is exact, according to the input dimension for (a) Theorem 1 and (b) Montúfar (2017).

We now build towards the proof of Theorem 1. For a given activation set S^l and a matrix W with n_l rows, let $\sigma_{S^l}(W)$ be the operation that zeroes out the rows of W that are inactive according to S^l. This represents the effect of the ReLUs. We say that a region is at layer l if it corresponds to an activation pattern $(S^1, \ldots, S^l)$ up to layer l. For a region $\mathcal{S}$ at layer $l - 1$, define $\bar{W}_{\mathcal{S}}^l := W^l \, \sigma_{S^{l-1}}(W^{l-1}) \cdots \sigma_{S^1}(W^1)$.

Each region $\mathcal{S}$ at layer $l - 1$ may be partitioned by a set of hyperplanes defined by the neurons of layer l. When viewed in the input space, these hyperplanes are the rows of $\bar{W}_{\mathcal{S}}^l x + b = 0$ for some b. To verify this, note that, if we recursively substitute out the hidden variables $h_{l-1}, \ldots, h_1$ from the original hyperplane $W^l h_{l-1} + b_l = 0$ following $\mathcal{S}$, the resulting weight matrix applied to x is $\bar{W}_{\mathcal{S}}^l$.

A central element in the proof of Theorem 1 is the dimension of the image of a linear region $\mathcal{S}$ under the DNN function $\mathbf{h}^l$ up to a layer l, which we denote by $\dim(\mathbf{h}^l(\mathcal{S}))$. For a fixed region $\mathcal{S}$ at layer l, that function is linear, and thus the dimension
$\dim(\mathbf{h}^l(\mathcal{S})) = \mathrm{rank}(\sigma_{S^l}(W^l) \cdots \sigma_{S^1}(W^1))$. This can be interpreted as the dimension of the space corresponding to $\mathcal{S}$ that the hyperplanes defined by $W^{l+1} h_l + b_{l+1} = 0$ effectively partitions. A key observation is that, once this dimension falls to a certain value, the regions contained in $\mathcal{S}$ at subsequent layers cannot recover to a higher dimension.

Zaslavsky (1975) showed that the maximal number of regions in $\mathbb{R}^d$ induced by an arrangement of m hyperplanes is at most $\sum_{j=0}^{d} \binom{m}{j}$. Moreover, this value is attained if

and only if the hyperplanes are in general position. The lemma below tightens this bound for a special case where the hyperplanes may not be in general position.

Lemma 4. *Consider m hyperplanes in $\mathbb{R}^d$ defined by the rows of $Wx + b = 0$. Then the number of regions induced by the hyperplanes is at most $\sum_{j=0}^{\mathrm{rank}(W)} \binom{m}{j}$.*

The proof is given in Appendix C. Its key idea is that it suffices to count regions within the row space of W. The next lemma brings Lemma 4 into our context.

Lemma 5. *The number of regions induced by the n_l neurons at layer l within a certain region $\mathcal{S}$ is at most $\sum_{j=0}^{\min\{n_l, \dim(\mathbf{h}^{l-1}(\mathcal{S}))\}} \binom{n_l}{j}$.*

Proof. The hyperplanes in a region $\mathcal{S}$ of the input space are given by the rows of $\bar{W}_{\mathcal{S}}^l x + b = 0$ for some b. By definition, the rank of $\bar{W}_{\mathcal{S}}^l$ is upper bounded by $\min\{\mathrm{rank}(W^l), \mathrm{rank}(\sigma_{S^{l-1}}(W^{l-1}) \cdots \sigma_{S^1}(W^1))\} = \min\{\mathrm{rank}(W^l), \dim(\mathbf{h}^{l-1}(\mathcal{S}))\}$. That is, $\mathrm{rank}(\bar{W}_{\mathcal{S}}^l) \leq \min\{n_l, \dim(\mathbf{h}^{l-1}(\mathcal{S}))\}$, and we apply Lemma 4. □

In the next lemma, we show that the dimension of the image of a region $\mathcal{S}$ can be bounded recursively in terms of the dimension of the image of the region containing $\mathcal{S}$ and the number of activated neurons defining $\mathcal{S}$.

Lemma 6. *Let $\mathcal{S}$ be a region at layer l and $\mathcal{S}'$ be the region at layer $l - 1$ that contains it. Then $\dim(\mathbf{h}^l(\mathcal{S})) \leq \min\{|S^l|, \dim(\mathbf{h}^{l-1}(\mathcal{S}'))\}$.*

Proof. $\dim(\mathbf{h}^l(\mathcal{S})) = \mathrm{rank}(\sigma_{S^l}(W^l) \cdots \sigma_{S^1}(W^1)) \leq \min\{\mathrm{rank}(\sigma_{S^l}(W^l)), \mathrm{rank}(\sigma_{S^{l-1}}(W^{l-1}) \cdots \sigma_{S^1}(W^1)) \leq \min\{|S^l|, \dim(\mathbf{h}^{l-1}(\mathcal{S}'))\}$. The last inequality comes from zeroed out rows not counting towards the matrix rank. □

We now have the ingredients to prove Theorem 1.

Proof of Theorem 1. As illustrated in Figure 2, the partitioning can be regarded as a sequential process: at each layer, we partition the regions obtained from the previous layer. When viewed in the input space, each region $\mathcal{S}$ obtained at layer $l - 1$ is potentially partitioned by n_l hyperplanes given by the rows of $\bar{W}_{\mathcal{S}}^l x + b = 0$ for some bias b. Some hyperplanes may fall outside the interior of $\mathcal{S}$ and have no effect.

With this process in mind, we recursively bound the number of subregions within a region. More precisely, we construct a recurrence $R(l, d)$ to upper bound the maximal number of regions attainable from partitioning a region with image of dimension d using units from layers $l, l+1, \ldots, L$. The base case is given by Lemma 5: $R(L, d) = \sum_{j=0}^{\min\{n_L, d\}} \binom{n_L}{j}$. Based on Lemma 6, the recurrence groups together regions with same activation set size $|S^l|$, as follows: $R(l, d) = \sum_{j=0}^{n_l} N_{n_l, d, j} R(l+1, \min\{j, d\})$ for all $l = 1, \ldots, L-1$.

Here, $N_{n_l,d,j}$ represents the maximum number of regions with $|S^l| = j$ from partitioning a space of dimension d with n_l hyperplanes. We bound this value next.

For each j, there are at most $\binom{n_l}{j}$ regions with $|S^l| = j$, as they can be viewed as subsets of n_l neurons of size j. In total, Lemma 5 states that there are at most $\sum_{j=0}^{\min\{n_l,d\}} \binom{n_l}{j}$ regions. If we allow these regions to have the highest $|S^l|$ possible, for each j from 0 to $\min\{n_l, d\}$ we have at most $\binom{n_l}{n_l-j} = \binom{n_l}{j}$ regions with $|S^l| = n_l - j$.

Therefore, we can write the recurrence as

$$R(l,d) = \begin{cases} \displaystyle\sum_{j=0}^{\min\{n_l,d\}} \binom{n_l}{j} R(l+1, \min\{n_l - j, d\}) \\ \qquad\qquad\qquad \text{if } 1 \leq l \leq L-1, \\ \displaystyle\sum_{j=0}^{\min\{n_L,d\}} \binom{n_L}{j} \\ \qquad\qquad\qquad \text{if } l = L. \end{cases}$$

The recurrence $R(1, n_0)$ can be unpacked to

$$\sum_{j_1=0}^{\min\{n_1,d_1\}} \binom{n_1}{j_1} \sum_{j_2=0}^{\min\{n_2,d_2\}} \binom{n_2}{j_2} \cdots \sum_{j_L=0}^{\min\{n_L,d_L\}} \binom{n_L}{j_L}$$

where $d_l = \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}\}$. This can be made more compact, resulting in the final expression.

The bound is tight when $L = 1$ since it becomes $\sum_{j=0}^{\min\{n_0,n_1\}} \binom{n_1}{j}$, which is the maximal number of regions of a single-layer network. $\qquad\square$

As a side note, Theorem 1 can be further tightened if the weight matrices are known to have small rank. The bound from Lemma 5 can be rewritten as $\sum_{j=0}^{\min\{\text{rank}(W^l),\dim(\mathbf{h}^{l-1}(\mathcal{S}))\}} \binom{n_l}{j}$ if we do not relax $\text{rank}(W^l)$ to n_l in the proof. The term $\text{rank}(W^l)$ follows through the proof of Theorem 1 and the index set J in the theorem becomes $\{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \leq j_l \leq \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}, \text{rank}(W^l)\} \forall l \geq 1\}$.

A key insight from Lemmas 5 and 6 is that the dimensions of the images of the regions are non-increasing as we move through the layers partitioning them. In other words, if at any layer the dimension of the image of a region becomes small, then that region will not be able to be further partitioned into a large number of regions. For instance, if the dimension of the image of a region falls to zero, then that region will never be further partitioned. This suggests that if we want to have many regions, we need to keep dimensions high. We use this idea in the next section to construct a DNN with many regions.

3.2. The Case of Dimension One

If the input dimension n_0 is equal to 1 and $n_l = n$ for all layers l, the upper bound presented in the previous section reduces to $(n+1)^L$. On the other hand, the lower bound given by Montúfar et al. (2014) becomes $n^{L-1}(n+1)$. It is then natural to ask: are either of these bounds tight? The answer is that the upper bound is tight in the case of $n_0 = 1$, assuming there are sufficiently many neurons.

Theorem 7. *Consider a deep rectifier network with L layers, $n_l \geq 3$ rectified linear units at each layer l, and an input of dimension 1. The maximal number of regions of this neural network is exactly $\prod_{l=1}^{L}(n_l + 1)$.*

The expression above is a simplified form of the upper bound from Theorem 1 in the case $n_0 = 1$.

The proof of this theorem in Appendix D has a construction with $n+1$ regions that replicate themselves as we add layers, instead of n as in Montúfar et al. (2014). That is motivated by an insight from the previous section: in order to obtain more regions, we want the dimension of the image of every region to be as large as possible. When $n_0 = 1$, we want all regions to have images with dimension one. This intuition leads to a new construction with one additional region that can be replicated with other strategies.

3.3. Lower Bounds on the Maximal Number of Linear Regions

Both the lower bounds from Montúfar et al. (2014) and from Arora et al. (2018) can be slightly improved, since their approaches are based on extending a 1-dimensional construction similar to the one in Section 3.2. We do both since they are not directly comparable: the former bound is in terms of the number of neurons in each layer and the latter is in terms of the total size of the network.

Theorem 8. *The maximal number of linear regions induced by a rectifier network with n_0 input units and L hidden layers with $n_l \geq 3n_0$ for all l is lower bounded by $\left(\prod_{l=1}^{L-1}\left(\lfloor \frac{n_l}{n_0}\rfloor + 1\right)^{n_0}\right)\sum_{j=0}^{n_0}\binom{n_L}{j}$.*

The proof of this theorem is in Appendix E. For comparison, the differences between the lower bound theorem (Theorem 5) from Montúfar et al. (2014) and the above theorem is the replacement of the condition $n_l \geq n_0$ by the more restrictive $n_l \geq 3n_0$, and of $\lfloor n_l/n_0 \rfloor$ by $\lfloor n_l/n_0 \rfloor + 1$.

Theorem 9. *For any $m \geq 1$ and $w \geq 2$, there exists a rectifier network with n_0 input units and L hidden layers of size $2m + w(L-1)$ that has $2\sum_{j=0}^{n_0-1}\binom{m-1}{j}(w+1)^{L-1}$ linear regions.*

The proof of this theorem is in Appendix F. The differences between Theorem 2.11(i) from Arora et al. (2018) and the above theorem is the replacement of w by $w + 1$. They

construct a $2m$-width layer with many regions and use a one-dimensional construction for the remaining layers.

4. An Upper Bound on the Number of Linear Regions for Maxout Networks

We now consider a deep neural network composed of maxout units. Given weights W_j^l for $j = 1, \ldots, k$, the output of a rank-k maxout layer l is given by

$$\mathbf{h}^l \quad = \quad \max\{W_1^l\mathbf{h}^{l-1} + b_1^l, \ldots, W_k^l\mathbf{h}^{l-1} + b_k^l\}$$

In terms of bounding number of regions, a major difference between the next result for maxout units and the previous one for ReLUs is that dimensionality plays less of a role, since neurons may no longer have an inactive state with zero output. Nevertheless, using techniques similar to the ones from Section 3.1, the following theorem can be shown (see Appendix G for the proof).

Theorem 10. *Consider a deep neural network with L layers, n_l rank-k maxout units at each layer l, and an input of dimension n_0. The maximal number of regions is at most*

$$\prod_{l=1}^{L} \sum_{j=0}^{d_l} \binom{\frac{k(k-1)}{2} n_l}{j}$$

where $d_l = \min\{n_0, n_1, \ldots, n_l\}$.

Asymptotically, if $n_l = n$ for all $l = 1, \ldots, L$, $n \geq n_0$, and $n_0 = O(1)$, then the maximal number of regions is at most $O((k^2 n)^{Ln_0})$.

5. Exact Counting of Linear Regions

If the input space $\mathbf{x} \in \mathbb{R}^{n_0}$ is bounded by minimum and maximum values along each dimension, or else if $\mathbf{x}$ corresponds to a polytope more generally, then we can define a mixed-integer linear formulation mapping polyhedral regions of $\mathbf{x}$ to the output space $\mathbf{y}$. The assumption that $\mathbf{x}$ is bounded and polyhedral is natural in most applications, where each value x_i has known lower and upper bounds (e.g., the value can vary from 0 to 1 for image pixels). Among other things, we can use this formulation to count the number of linear regions.

In the formulation that follows, we use continuous variables to represent the input $\mathbf{x}$, which we can also denote as $\mathbf{h}^0$, the output of each neuron i in layer l as h_i^l, and the output $\mathbf{y}$ as $\mathbf{h}^{L+1}$. To simplify the representation, we lift this formulation to a space that also contains the output of a complementary set of neurons, each of which is active when the corresponding neuron is not. Namely, for each neuron i in layer l we also have a variable $\overline{h}_i^l := \max\{0, -W_i^l h^{l-1} - b_i^l\}$. We use binary variables of the form z_i^l to denote if each

neuron i in layer l is active or else if the complement is. Finally, we assume M to be a sufficiently large constant.

For a given neuron i in layer l, the following set of constraints maps the input to the output:

$$W_i^l h^{l-1} + b_i^l = h_i^l - \overline{h}_i^l \tag{1}$$
$$h_i^l \leq M z_i^l \tag{2}$$
$$\overline{h}_i^l \leq M(1 - z_i^l) \tag{3}$$
$$h_i^l \geq 0 \tag{4}$$
$$\overline{h}_i^l \geq 0 \tag{5}$$
$$z_i^l \in \{0, 1\} \tag{6}$$

Theorem 11. *Provided that $|W_i^l h^{l-1} + b_i^l| \leq M$ for any possible value of h^{l-1}, a formulation with the set of constraints (1)–(6) for each neuron of a rectifier network is such that a feasible solution with a fixed value for x yields the output y of the neural network.*

Proof. It suffices to prove that the constraints for each neuron map the input to the output in the same way that the neural network would. If $W_i^l \mathbf{h}^{l-1} + b_i^l > 0$, it follows that $h_i^l - \overline{h}_i^l > 0$ according to (1). Since both variables are non-negative due to (4) and (5) whereas one is non-positive due to (2), (3), and (6), then $z_i^l = 1$ and $h_i^l = \max\{0, W_i^l \mathbf{h}^{l-1} + b_i^l\}$. If $W_i^l \mathbf{h}^{l-1} + b_i^l < 0$, then it similarly follows that $h_i^l - \overline{h}_i^l < 0$, $z_i^l = 0$, and thus $\overline{h}_i^l = \min\{0, W_i^l \mathbf{h}^{l-1} + b_i^l\}$. If $W_i^l \mathbf{h}^{l-1} + b_i^l = 0$, then either $h_i^l = 0$ or $\overline{h}_i^l = 0$ due to constraints (4) to (6) whereas (1) implies that $\overline{h}_i^l = 0$ or $h_i^l = 0$, respectively. In this case, the value of z_i^l is arbitrary but irrelevant. $\square$

A systematic method to count integer solutions is the one-tree approach (Danna et al., 2007), which resumes the search after an optimal solution has been found using the same branch-and-bound tree. This method also allows for counting feasible solutions within a threshold of the optimal value. Note that in constraints (1)–(6), the variables z_i^l can be either 0 or 1 when they lie on the activation boundary, whereas we want to consider a neuron active only when its output is strictly positive. This discrepancy may cause double-counting when activation boundaries overlap. We can address that by defining an objective function that maximizes the minimum output f of an active neuron, which is positive in the non-degenerate cases that we want to count. We state this formulation for rectifier networks as follows:

$$\max \quad f$$
$$\text{s.t. } (1)–(6) \qquad\qquad \forall \text{ neuron } i \text{ in layer } l \quad (\mathcal{P})$$
$$\quad f \leq h_i^l + (1 - z_i^l)M \quad \forall \text{ neuron } i \text{ in layer } l$$
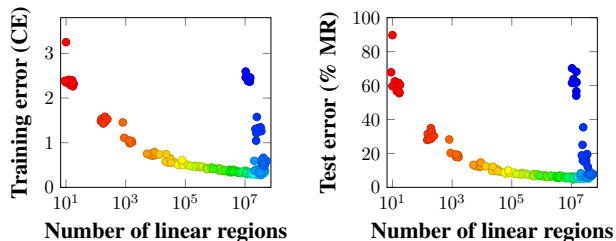$$\quad x \in X$$

Figure 6. *Number of linear regions versus cross-entropy error on the training set, and misclassification rate on the testing set. The colors represent the number of neurons in each of the first two layers. The color gradient from red to blue corresponds to increasing the size of the first layer and decreasing the size of the second.*
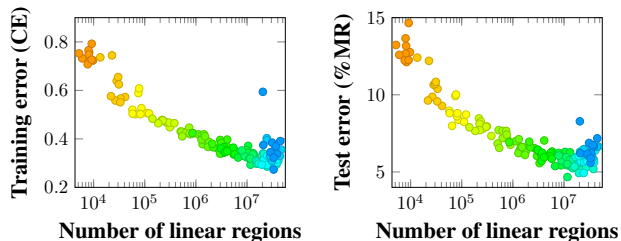

Figure 7. *Scatter plot for the number of linear regions versus training and testing errors on DNNs with minimum width at least 4. The colors follow the same convention used in Figure 6.*

A discusssion on the choice for the M constants can be found in Appendix H. In addition, we discuss the mixed-representability of DNNs in Appendix I, the theory for unrestricted inputs in Appendix J, and a mixed-integer formulation for maxout networks in Appendix K, respectively.

6. Experiments

We perform an experiment to count linear regions of small-sized networks with ReLU activation units on the MNIST benchmark dataset (LeCun et al., 1998). In this experiment, we train rectifier networks with two hidden layers summing up to 22 neurons. We train 10 networks for each configuration for 20 epochs or training steps, and we count all linear regions within $0 \leq \mathbf{x} \leq 1$. The counting code is written in C++ (gcc 4.8.4) using CPLEX Studio 12.8 as a solver and ran in Ubuntu 14.04.4 on a machine with 40 Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz processors and 132 GB of RAM. The runtimes for counting different configuration can be found in Appendix L.

Figure 1 shows the average, minimum, and maximum number of linear regions for each configuration of these networks. The plot also contains the corresponding upper bounds for each configuration from Theorem 1 and those from Montúfar et al. (2014) and Montúfar (2017), which are the first and the tightest bounds from prior work respectively. Note that upper bound from Theorem 1 is tighter.

Figure 6 shows how the number of regions for each trained DNN compares with Cross-Entropy (CE) error on training set, and Misclassification Rate (MR) on testing set. If an intermediate layer in a network has only 1 or 2 neurons, it is natural to expect very high training and test errors. We discard such DNNs with a layer of small width in Figure 7.

7. Discussion

Our ReLU upper bound indicates that small widths in early layers cause a bottleneck effect on the number of regions. If

we reduce the width of an early layer, the dimension of the image of the linear regions become irrecoverably smaller throughout the network and the regions will not be able to be partitioned as much. This intuition allows us to develop a 1-dimensional construction with the maximal number of regions by eliminating a zero-dimensional bottleneck. In our experiment, we validate the bottleneck effect by observing that the actual number of linear regions is asymmetric when one layer is increased and another decreased in size.

An unexpected consequence of one of our results is that shallow networks can attain more linear regions when the input dimensions exceed the number of neurons. This complements prior work, which has not considered large input dimensions, a common case in practice.

We also observe in Figure 5 that the depth that maximizes the upper bound from Theorem 1 increases with the number of units and decreases with the size of the input. It would be interesting to investigate if this also happens with respect to the actual number of regions, in which case the depth could be chosen according to those parameters.

However, it would be possible that DNNs configurations with large number of linear regions do not generalize well if there are so many regions that each training point can be singled out in a different region, in particular if regions with similar labels are unlikely to be compositionally related.

Nevertheless, we have initial evidence that training and classification accuracies relate to the number of regions for configurations where no layer is too small, hence suggesting that the number of regions can be a metric for comparing similar DNN configurations. However, in the cases where one layer is too small, we have also observed that the number of regions can be large and do not reflect the capacity of the DNN. We hypothesize that the presence of low dimensionality negatively affects the accuracy of the network, even when the number of regions is large because of other layers. This indicates that potentially more insights could be gained from investigating the shape of linear regions.

Acknowledgements

References

Anthony, M. and Bartlett, P. Neural network learning: Theoretical foundations. 1999.

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

Balas, E. Disjunctive programming. *Annals of Discrete Mathematics*, (5):3–51, 1979.

Balas, E., Ceria, S., and Cornuéjols, G. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58:295–324, 1993.

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*, 2014.

Camm, J. D., Raturi, A. S., and Tsubakitani, S. Cutting big M down to size. *Interfaces*, 20(5):61–66, 1990.

Cheng, C.-H., Nührenberg, G., and Ruess, H. Maximum resilience of artificial neural networks. In *ATVA*, pp. 251–268, 2017.

Ciresan, D., Meier, U., Masci, J., and Schmidhuber, J. Multi column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Danna, E., Fenelon, M., Gu, Z., and Wunderling, R. Generating multiple solutions for mixed integer programming problems. In *IPCO*, pp. 280–294. 2007.

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

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

Fourier, J. Solution dune question particuliére du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société Philomatique de Paris*, pp. 317–319, 1826.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. In *ICML*, 2013.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hinton, G., Deng, L., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 2012.

Jeroslow, R. Representability in mixed integer programming, I: Characterization results. *Discrete Applied Mathematics*, 17(3):223 – 243, 1987.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Maass, W., Schnitger, G., and Sontag, E. A comparison of the computational power of sigmoid and boolean threshold circuits. *Theoretical Advances in Neural Computation and Learning*, pp. 127–151, 1994.

Mhaskar, H., Liao, Q., and Poggio, T. A. Learning real and boolean functions: When is deep better than shallow. *CoRR*, abs/1603.00988, 2016.

Montúfar, G. Notes on the number of linear regions of deep neural networks. In *SampTA*, 2017.

Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

Pan, X. and Srikumar, V. Expressiveness of rectifier networks. In *ICML*, 2016.

Pascanu, R., Montúfar, G., and Bengio, Y. On the number of response regions of deep feedforward networks with piecewise linear activations. In *ICLR*, 2014.

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

Stirling, J. *Methodus Differentialis sive Tractatus de Summatione et Interpolatione Serierum Infinitarum*. G. Strahan, London, 1730.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Telgarsky, M. Representation benefits of deep feedforward networks. *CoRR*, abs/1509.08101, 2015.

Zaslavsky, T. *Facing up to arrangements: face-count formulas for partitions of space by hyperplanes*. American Mathematical Society, 1975.