Appendices

Most of the proofs for theorems and lemmas associated with the upper and lower bounds on the linear regions are provided below. We also discuss counting on unrestricted input and the mixed-integer formulation for maxout networks.

A. Analysis of the Bound from Theorem 1

In this section, we present properties of the upper bound for the number of regions of a rectifier network from Theorem 1. Denote the bound by $B(n_0, n_1, \ldots, n_L)$, where n_0 is the input dimension and $n_1, \ldots, n_L$ are the widths of layers 1 through L of the network. That is,

$$B(n_0, n_1, \ldots, n_L) := \sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l}$$

Instead of expressing J as in Theorem 1, we rearrange it to a more convenient form for the proofs in this section:

$$\begin{aligned} J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : & j_l + j_k \leq n_k \ \forall k = 1, \ldots, l-1 \ \forall l = 2, \ldots, L \\ & j_l \leq n_0 \ \forall l = 1, \ldots, L \\ & 0 \leq j_l \leq n_l \ \forall l = 1, \ldots, L\}. \end{aligned}$$

Note that whenever we assume $n_0 \geq \max\{n_1, \ldots, n_l\}$, then the bound inequality for n_0 becomes redundant and can be removed.

Some of the results have implications in terms of the exact maximal number of regions. We denote it by $R(n_0, n_1, \ldots, n_L)$, following the same notation above.

Moreover, the following lemma is useful throughout the section.

Lemma 12.

$$\sum_{j=0}^{k} \binom{n_1 + \ldots + n_L}{j} = \sum_{\substack{j_1+\ldots+j_L\leq k \\ 0\leq j_l\leq n_l \ \forall l}} \binom{n_1}{j_1}\binom{n_2}{j_2}\cdots\binom{n_L}{j_L}.$$

Proof. The result comes from taking a generalization of Vandermonde's identity and adding the summation of j from 0 to k as above. □

We first examine some properties related to 2-layer networks. The proposition below characterizes the bound when $L = 2$ for large input dimensions.

Proposition 13. *Consider a 2-layer network with widths n_1, n_2 and input dimension $n_0 \geq n_1$ and $n_0 \geq n_2$. Then*

$$B(n_0, n_1, n_2) = \sum_{j=0}^{n_1} \binom{n_1 + n_2}{j}$$

If $n_0 < n_1$ or $n_0 < n_2$, the above holds with inequality: $B(n_0, n_1, n_2) \leq \sum_{j=0}^{n_1} \binom{n_1+n_2}{j}$.

Proof. If $n_0 \geq n_1$ and $n_0 \geq n_2$, the bound inequalities for n_0 in the index set J become redundant. By applying Lemma 12, we obtain

$$B(n_0, n_1, n_2) = \sum_{\substack{0\leq j_1+j_2\leq n_1 \\ 0\leq j_l\leq n_l \ \forall l}} \binom{n_1}{j_1}\binom{n_2}{j_2} = \sum_{j=0}^{n_1} \binom{n_1 + n_2}{j}.$$

If $n_0 < n_1$ or $n_0 < n_2$, then its index set J is contained by the one above, and thus the first equal sign above becomes a less-or-equal sign. □

Recall that the expression on the right-hand side of Proposition 13 is equal to the maximal number of regions of a single-layer network with $n_1 + n_2$ ReLUs and input dimension n_1, as discussed in Section 2. Hence, the proposition implies that for large input dimensions, a two-layer network has no more regions than a single-layer network with the same number of neurons, as formalized below.

Corollary 14. *Consider a 2-layer network with widths $n_1, n_2 \geq 1$ and input dimension $n_0 \geq n_1$ and $n_0 \geq n_2$. Then $R(n_0, n_1, n_2) \leq R(n_0, n_1 + n_2)$.*

Moreover, this inequality is strict when $n_0 > n_1$.

Proof. This is a direct consequence of Proposition 13:

$$R(n_0, n_1, n_2) \leq B(n_0, n_1, n_2) = \sum_{j=0}^{n_1} \binom{n_1 + n_2}{j} \leq \sum_{j=0}^{n_0} \binom{n_1 + n_2}{j} = R(n_0, n_1 + n_2).$$

Note that if $n_0 > n_1$, then the second inequality can be turned into a strict inequality. □

The next proposition illustrates the bottleneck effect for two layers. It states that for large input dimensions, moving a neuron from the second layer to the first strictly increases the bound. This proposition is stated more informally in the main text.

Proposition 2. *Consider a 2-layer network with widths n_1, n_2 and input dimension $n_0 \geq n_1 + 1$ and $n_0 \geq n_2 + 1$. Then $B(n_0, n_1 + 1, n_2) > B(n_0, n_1, n_2 + 1)$.*

Proof. By Proposition 13,

$$B(n_0, n_1 + 1, n_2) = \sum_{j=0}^{n_1+1} \binom{(n_1 + 1) + n_2}{j} > \sum_{j=0}^{n_1} \binom{n_1 + (n_2 + 1)}{j} = B(n_0, n_1, n_2 + 1).$$

□

The assumption that n_0 must be large is required for the above proposition; otherwise, the input itself may create a bottleneck with respect to the second layer as we decrease its size. Note that the bottleneck affects all subsequent layers, not only the layer immediately after it.

However, it is not true that moving neurons to earlier layers always increases the bound. For instance, with three layers, $B(4, 3, 2, 1) = 47 > 46 = B(4, 4, 1, 1)$.

In the remainder of this section, we consider deep networks of equal widths n. The next proposition can be viewed as an extension of Proposition 13 for multiple layers. It states that for a network with widths and input dimension n and at least 4 layers, if we halve the number of layers and redistribute the neurons so that the widths become $2n$, then the bound increases. In other words, if we assume the bound to be close to the maximal number of regions, it suggests that making a deep network shallower allows for more regions when the input dimension is equal to the width.

Proposition 15. *Consider a $2L$-layer network with equal widths n and input dimension $n_0 = n$. Then*

$$B(n, \underbrace{n, \ldots, n}_{2L \text{ times}}) \leq B(n, \underbrace{2n, \ldots, 2n}_{L \text{ times}}).$$

This inequality is met with equality when $L = 1$ and strict inequality when $L \geq 2$.

Proof. When $n_0 = n$, the inequalities $j_l \leq \min\{n_0, 2n - j_1, \ldots, 2n - j_{l-1}, 2n\}$ appearing in J (in the form presented in Theorem 1) can be simplified to $j_l \leq n$. Therefore, using Lemma 12, the bound on the right-hand side becomes

$$B(n, \underbrace{2n, \ldots, 2n}_{L \text{ times}}) = \sum_{j_1=0}^{n} \sum_{j_2=0}^{n} \cdots \sum_{j_L=0}^{n} \prod_{l=1}^{L} \binom{2n}{j_l} = \left(\sum_{j=0}^{n} \binom{2n}{j} \right)^L = \left(\sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \binom{n}{j_1} \binom{n}{j_2} \right)^L$$

$$\geq \sum_{(j_1, \ldots, j_{2L}) \in J} \prod_{l=1}^{2L} \binom{n}{j_l} = B(n, \underbrace{n, \ldots, n}_{2L \text{ times}}).$$

where J above is the index set from Theorem 1 applied to $n_0 = n_l = n$ for all $l = 1, \ldots, 2L$. Note that we can turn the inequality into equality when $L = 1$ (also becoming a consequence of Proposition 13) and into strict inequality when $L \geq 2$. □

Next, we provide an upper bound that is independent of n_0 based on Theorem 1.

Corollary 16. *Consider an L-layer network with equal widths n and any input dimension $n_0 \geq 0$.*

$$B(n_0, n, \ldots, n) \leq 2^{Ln} \left(\frac{1}{2} + \frac{1}{2\sqrt{\pi n}} \right)^{L/2} \sqrt{2}$$

Proof. Since we are deriving an upper bound, we can assume $n_0 \geq n$, as the bound is nondecreasing on n_0. We first assume that L is even. We relax some of the constraints of the index set J from Theorem 1 and apply Vandermonde's identity on each pair:

$$\begin{aligned} B(n_0, n, \ldots, n) &\leq \sum_{j_1=0}^{n} \sum_{j_2=0}^{n-j_1} \binom{n}{j_1}\binom{n}{j_2} \sum_{j_3=0}^{n} \sum_{j_4=0}^{n-j_3} \binom{n}{j_3}\binom{n}{j_4} \cdots \sum_{j_{L-1}=0}^{n} \sum_{j_L=0}^{n-j_{L-1}} \binom{n}{j_{L-1}}\binom{n}{j_L} \\ &= \left(\sum_{j=0}^{n} \binom{2n}{j} \right)^{L/2} = \left(\frac{2^{2n} + \binom{2n}{n}}{2} \right)^{L/2} \leq \left(\frac{2^{2n} + \frac{2^{2n}}{\sqrt{\pi n}}}{2} \right)^{L/2} \\ &= 2^{Ln} \left(\frac{1}{2} + \frac{1}{2\sqrt{\pi n}} \right)^{L/2}. \end{aligned}$$

The bound on $\binom{2n}{n}$ is a direct application of Stirling's approximation (Stirling, 1730). If L is odd, then we can write

$$\begin{aligned} B(n_0, n, \ldots, n) &\leq \left(\sum_{j=0}^{n} \binom{2n}{j} \right)^{(L-1)/2} \left(\sum_{j=0}^{n} \binom{n}{j} \right) = \left(\sum_{j=0}^{n} \binom{2n}{j} \right)^{L/2} \frac{2^n}{\left(\sum_{j=0}^{n} \binom{2n}{j} \right)^{1/2}} \\ &\leq \left(\sum_{j=0}^{n} \binom{2n}{j} \right)^{L/2} \frac{2^n}{(2^{2n}/2)^{1/2}} = \left(\sum_{j=0}^{n} \binom{2n}{j} \right)^{L/2} \sqrt{2} \\ &\leq 2^{Ln} \left(\frac{1}{2} + \frac{1}{2\sqrt{\pi n}} \right)^{L/2} \sqrt{2} \end{aligned}$$

where the last inequality is analogous to the even case. Hence, the result follows. □

We now use Corollary 16 to show that a shallow network can attain more linear regions than a deeper one of the same size when the input dimension n_0 exceeds the total number of neurons. This result is stated in the main text.

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

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

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

Proof. Note first that if $n_0 \geq Ln$, then $R(n_0, Ln) = \sum_{j=0}^{Ln} \binom{Ln}{j} = 2^{Ln}$.

The inequality can be derived from Proposition 16, since $\left(\frac{1}{2} + \frac{1}{2\sqrt{\pi n}} \right)^{L/2} \sqrt{2} < 1$ when $L \geq 3$ and $n \geq 1$, and thus $R(n_0, n, \ldots, n) \leq B(n_0, n, \ldots, n) < 2^{Ln} = R(n_0, Ln)$. The same holds when $L = 2$: as noted in the proof of Proposition 16, we may discard the factor of $\sqrt{2}$ when L is even, and $\left(\frac{1}{2} + \frac{1}{2\sqrt{\pi n}} \right)^{L/2} < 1$ for $L = 2$ and $n \geq 1$.

Proposition 16 also implies that

$$\frac{R(n_0, n, \ldots, n)}{R(n_0, Ln)} \leq \frac{B(n_0, n, \ldots, n)}{2^{Ln}} \leq \left(\frac{1}{2} + \frac{1}{2\sqrt{\pi n}}\right)^{L/2} \sqrt{2}.$$

Since the base of the first term of the above expression is less than 1 for all $n \geq 1$ and $\sqrt{2}$ is a constant, the ratio goes to 0 as L goes to infinity. □

B. Exponential Maximal Number of Regions When Input Dimension is Large

Proposition 17. *Consider an L-layer rectifier network with equal widths n and input dimension $n_0 \geq n/3$. Then the maximal number of regions is $\Omega(2^{\frac{2}{3}Ln})$.*

Proof. It suffices to show that a lower bound such as the one from Theorem 8 grows exponentially large. For simplicity, we consider the lower bound $(\prod_{l=1}^{L}(\lfloor n_l/n_0 \rfloor + 1))^{n_0}$, which is the bound obtained before the last tightening step in the proof of Theorem 8 (see Appendix E).

Note that replacing n_0 in the above expression by a value n_0' smaller than the input dimension still yields a valid lower bound. This holds because increasing the input dimension of a network from n_0' to n_0 cannot decrease its maximal number of regions.

Choose $n_0' = \lfloor n/3 \rfloor$, which satisfies $n_0' \leq n_0$ and the condition $n \geq 3n_0'$ of Theorem 8. The lower bound can be expressed as $(\lfloor n/\lfloor n/3 \rfloor \rfloor + 1)^{L\lfloor n/3 \rfloor} \geq 4^{L\lfloor n/3 \rfloor}$. This implies that the maximal number of regions is $\Omega(2^{\frac{2}{3}Ln})$. □

C. Proof of Lemma 4

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 $\sum_{j=0}^{\operatorname{rank}(W)} \binom{m}{j}$.*

Proof. Consider the row space $\mathcal{R}(W)$ of W, which is a subspace of $\mathbb{R}^d$ of dimension $\operatorname{rank}(W)$. We show that the number of regions $N_{\mathbb{R}^d}$ in $\mathbb{R}^d$ is equal to the number of regions $N_{\mathcal{R}(W)}$ in $\mathcal{R}(W)$ induced by $Wx + b = 0$ restricted to $\mathcal{R}(W)$. This suffices to prove the lemma since $\mathcal{R}(W)$ has at most $\sum_{j=0}^{\operatorname{rank}(W)} \binom{m}{j}$ regions according to Zaslavsky's theorem.

Since $\mathcal{R}(W)$ is a subspace of $\mathbb{R}^d$, it directly follows that $N_{\mathcal{R}(W)} \leq N_{\mathbb{R}^d}$. To show the converse, we apply the orthogonal decomposition theorem from linear algebra: any point $\bar{x} \in \mathbb{R}^d$ can be expressed uniquely as $\bar{x} = \hat{x} + y$, where $\hat{x} \in \mathcal{R}(W)$ and $y \in \mathcal{R}(W)^{\perp}$. Here, $\mathcal{R}(W)^{\perp} = \operatorname{Ker}(W) := \{y \in \mathbb{R}^d : Wy = 0\}$, and thus $W\bar{x} = W\hat{x} + Wy = W\hat{x}$. This means $\bar{x}$ and $\hat{x}$ lie on the same side of each hyperplane of $Wx = b$ and thus belong to the same region. In other words, given any $\bar{x} \in \mathbb{R}^d$, its region is the same one that $\hat{x} \in \mathcal{R}(W)$ lies in. Therefore, $N_{\mathbb{R}^d} \leq N_{\mathcal{R}(W)}$. Hence, $N_{\mathbb{R}^d} = N_{\mathcal{R}(W)}$ and the result follows. □

D. Proof of Theorem 7

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 $\prod_{l=1}^{L}(n_l + 1)$.*

Proof. Section 3 provides us with a helpful insight to construct an example with a large number of regions. It tells us that we want regions to have images with large dimension in general. In particular, regions with images of dimension zero cannot be further partitioned. This suggests that the one-dimensional construction from Montúfar et al. (2014) can be improved, as it contains n regions with images of dimension one and 1 region with image of dimension zero. This is because all ReLUs point to the same direction as depicted in Figure 8, leaving one region with an empty activation pattern.

Our construction essentially increases the dimension of the image of this region from zero to one. This is done by shifting the neurons forward and flipping the direction of the third neuron, as illustrated in Figure 8. We assume $n \geq 3$.

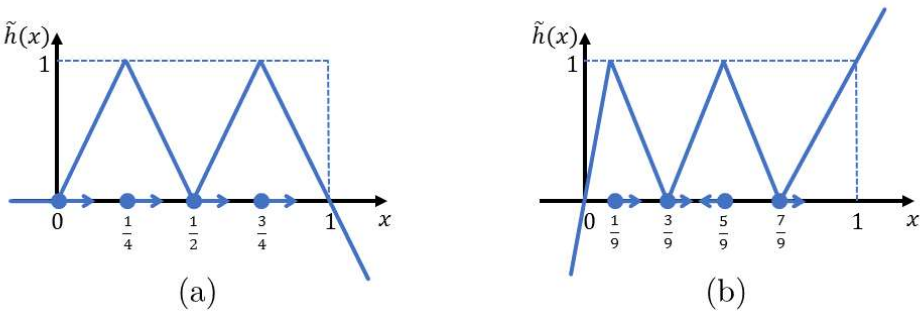

Figure 8. (a) The 1D construction from Montúfar et al. (2014). All units point to the right, leaving a region with image of dimension zero before the origin. (b) The 1D construction described in this section. Within the interval $[0, 1]$ there are five regions instead of the four in (a).

We review the intuition behind the construction strategy from Montúfar et al. (2014). They construct a linear function $\tilde{h} : \mathbb{R} \to \mathbb{R}$ with a zigzag pattern from $[0, 1]$ to $[0, 1]$ that is composed of n ReLUs. More precisely, $\tilde{h}(x) = (1, -1, 1, \ldots, \pm 1)^\top (h_1(x), h_2(x), \ldots, h_n(x))$, where $h_i(x)$ for $i = 1, \ldots, n$ are ReLUs. This linear function can be absorbed in the preactivation function of the next layer.

The zigzag pattern allows it to replicate in each slope a scaled copy of the function in the domain $[0, 1]$. Figure 9 shows an example of this effect. Essentially, when we compose $\tilde{h}$ with itself, each linear piece in $[t_1, t_2]$ such that $\tilde{h}(t_1) = 0$ and $\tilde{h}(t_2) = 1$ maps the entire function $\tilde{h}$ to the interval $[t_1, t_2]$, and each piece such that $\tilde{h}(t_1) = 1$ and $\tilde{h}(t_2) = 2$ does the same in a backward manner.

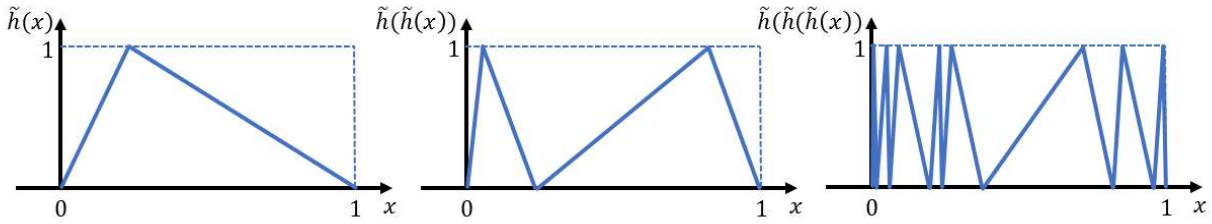

Figure 9. A function with a zigzag pattern composed with itself. Note that the entire function is replicated within each linear region, up to a scaling factor.

In our construction, we want to use n ReLUs to create $n + 1$ regions instead of n. In other words, we want the construct this zigzag pattern with $n + 1$ slopes. In order to do that, we take two steps to give ourselves more freedom. First, observe that we only need each linear piece to go from zero to one or one to zero; that is, the construction works independently of the length of each piece. Therefore, we turn the breakpoints into parameters $t_1, t_2, \ldots, t_n$, where $0 < t_1 < t_2 < \ldots < t_n < 1$. Second, we add sign and bias parameters to the function $\tilde{h}$. That is, $\tilde{h}(x) = (s_1, s_2, \ldots, s_n)^\top (h_1(x), h_2(x), \ldots, h_n(x)) + d$, where $s_i \in \{-1, +1\}$ and d are parameters to be set. Here, $h_i(x) = \max\{0, \tilde{w}_i x + \tilde{b}_i\}$ since it is a ReLU.

We define $w_i = s_i \tilde{w}_i$ and $b_i = s_i \tilde{b}_i$, which are the weights and biases we seek in each interval to form the zigzag pattern. The parameters s_i are needed because the signs of $\tilde{w}_i$ cannot be arbitrary: it must match the directions the ReLUs point towards. In particular, we need a positive slope ($\tilde{w}_i > 0$) if we want i to point right, and a negative slope ($\tilde{w}_i < 0$) if we want i to point left. Hence, without loss of generality, we do not need to consider the s_i's any further since they will be directly defined from the signs of the w_i's and the directions. More precisely, $s_i = 1$ if $w_i \geq 0$ and $s_i = -1$ otherwise for $i = 1, 2, 4, \ldots, n$, and $s_3 = -1$ if $w_3 \geq 0$ and $s_3 = 1$ otherwise.

To summarize, our parameters are the weights w_i and biases b_i for each ReLU, a global bias d, and the breakpoints

$0 < t_1 < \ldots < t_n < 1$. Our goal is to find values for these parameters such that each piece in the function $\tilde{h}$ with domain in $[0, 1]$ is linear from zero to one or one to zero.

More precisely, if the domain is $[s, t]$, we want each linear piece to be either $\frac{1}{t-s}x - \frac{s}{t-s}$ or $-\frac{1}{t-s}x + \frac{t}{t-s}$, which define linear functions from zero to one and from one to zero respectively. Since we want a zigzag pattern, the former should happen for the interval $[t_i, t_{i-1}]$ when i is odd and the latter should happen when i is even.

There is one more set of parameters that we will fix. Each ReLU corresponds to a hyperplane, or a point in dimension one. In fact, these points are the breakpoints $t_1, \ldots, t_n$. They have directions that define for which inputs the neuron is activated. For instance, if a neuron h_i points to the right, then the neuron $h_i(x)$ outputs zero if $x \leq t_i$ and the linear function $w_i x + b_i$ if $x > t_i$.

As previously discussed, in our construction all neurons point right except for the third neuron h_3, which points left. This is to ensure that the region before t_1 has one activated neuron instead of zero, which would happen if all neurons pointed left. However, although ensuring every region has images of dimension one is necessary to reach the bound, not every set of directions yields valid weights. These directions are chosen so that they admit valid weights.

The directions of the neurons tells us which neurons are activated in each region. From left to right, we start with h_3 activated, then we activate h_1 and h_2 as we move forward, we deactivate h_3, and finally we activate $h_4, \ldots, h_n$ in sequence. This yields the following system of equations, where t_{n+1} is defined as 1 for simplicity:

$$w_3 x + (b_3 + d) = \frac{1}{t_1} x \tag{R_1}$$

$$(w_1 + w_3)\, x + (b_1 + b_3 + d) = -\frac{1}{t_2 - t_1} x + \frac{t_2}{t_2 - t_1} \tag{R_2}$$

$$(w_1 + w_2 + w_3)\, x + (b_1 + b_2 + b_3 + d) = \frac{1}{t_3 - t_2} x - \frac{t_2}{t_3 - t_2} \tag{R_3}$$

$$(w_1 + w_2)\, x + (b_1 + b_2 + d) = -\frac{1}{t_4 - t_3} x + \frac{t_4}{t_4 - t_3} \tag{R_4}$$

$$\left(w_1 + w_2 + \sum_{j=4}^{i-1} w_j\right) x + \left(b_1 + b_2 + \sum_{j=4}^{i-1} b_j + d\right) = \begin{cases} \frac{1}{t_i - t_{i-1}} x - \frac{t_{i-1}}{t_i - t_{i-1}} & \text{if } i \text{ is odd} \\ -\frac{1}{t_i - t_{i-1}} x + \frac{t_i}{t_i - t_{i-1}} & \text{if } i \text{ is even} \end{cases} \tag{R_i}$$

for all $i = 5, \ldots, n+1$

It is left to show that there exists a solution to this system of linear equations such that $0 < t_1 < \ldots < t_n < 1$.

First, note that all of the biases $b_1, \ldots, b_n, d$ can be written in terms of $t_1, \ldots, t_n$. Note that if we subtract (R_4) from (R_3), we can express b_3 in terms of the t_i variables. The remaining equations become triangular, and therefore given any values for t_i's we can back-substitute the remaining bias variables.

The same subtraction yields w_3 in terms of t_i's. However, both (R_1) and $(R_3) - (R_4)$ define w_3 in terms of the t_i variables, so they must be the same:

$$\frac{1}{t_1} = \frac{1}{t_3 - t_2} + \frac{1}{t_4 - t_3}.$$

If we find values for t_i's satisfying this equation and $0 < t_1 < \ldots < t_n < 1$, all other weights can be obtained by back-substitution since eliminating w_3 yields a triangular set of equations.

In particular, the following values are valid: $t_1 = \frac{1}{2n+1}$ and $t_i = \frac{2i-1}{2n+1}$ for all $i = 2, \ldots, n$. The remaining weights and biases can be obtained as described above, which completes the desired construction.

As an example, a construction with four units is depicted in Figure 8. Its breakpoints are $t_1 = \frac{1}{9}$, $t_2 = \frac{3}{9}$, $t_3 = \frac{5}{9}$, and $t_4 = \frac{7}{9}$. Its ReLUs are $h_1(x) = \max\{0, -\frac{27}{2}x + \frac{3}{2}\}$, $h_2(x) = \max\{0, 9x - 3\}$, $h_3(x) = \max\{0, 9x - 5\}$, and $h_4(x) = \max\{0, 9x\}$. Finally, $\tilde{h}(x) = (-1, 1, -1, 1)^\top (h_1(x), h_2(x), h_3(x), h_4(x)) + 5$.

□

E. Proof of Theorem 8

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*

$$\left(\prod_{l=1}^{L-1}\left(\left\lfloor \frac{n_l}{n_0}\right\rfloor + 1\right)^{n_0}\right)\sum_{j=0}^{n_0}\binom{n_L}{j}.$$

Proof. We follow the proof of Theorem 5 from (Montúfar et al., 2014) except that we use a different 1-dimensional construction. The main idea of the proof is to organize the network into n_0 independent networks with input dimension 1 each and apply the 1-dimensional construction to each individual network. In particular, for each layer l we assign $\lfloor n_l/n_0 \rfloor$ ReLUs to each network, ignoring any remainder units. In (Montúfar et al., 2014), each of these networks have at least $\prod_{l=1}^{L}\lfloor n_l/n_0\rfloor$ regions. We instead use Theorem 7 to attain $\prod_{l=1}^{L}(\lfloor n_l/n_0\rfloor + 1)$ regions in each network.

Since the networks are independent from each other, the number of activation patterns of the compound network is the product of the number of activation patterns of each of the n_0 networks. Hence, the same holds for the number of regions. Therefore, the number of regions of this network is at least $(\prod_{l=1}^{L}(\lfloor n_l/n_0\rfloor + 1))^{n_0}$.

In addition, we can replace the last layer by a function representing an arrangement of n_L hyperplanes in general position that partitions $(0,1)^{n_0}$ into $\sum_{j=0}^{n_0}\binom{n_L}{j}$ regions. This yields the lower bound of $\prod_{l=1}^{L-1}(\lfloor n_l/n_0\rfloor + 1)^{n_0}\sum_{j=0}^{n_0}\binom{n_L}{j}$.

□

F. Proof of Theorem 9

Theorem 9. *For any values of $m \geq 1$ 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.*

Proof. Theorem 6.1 and Lemma 6.2 in Arora et al. (2018) imply that for any $m \geq 1$, we can construct a layer representing a function from $\mathbb{R}^n$ to $\mathbb{R}$ with $2m$ ReLUs that has $2\sum_{j=0}^{n_0-1}\binom{m-1}{j}$ regions. Consider the network where this layer is the first one and the remaining layers are the one-dimensional layers from Theorem 7, each of size w. Then this network has size $2m + w(L-1)$ and $2\sum_{j=0}^{n_0-1}\binom{m-1}{j}(w+1)^{L-1}$ regions. □

G. Proof of Theorem 10

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 of this neural network 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 $O((k^2 n)^{Ln_0})$.

Proof. We denote by W_j^l the $n_l \times n_{l-1}$ matrix where the rows are given by the j-th weight vectors of each rank-k maxout unit at layer l, for $j = 1, \ldots, k$. Similarly, b_j^l is the vector composed of the j-th biases at layer l.

In the case of maxout, an activation pattern $\mathcal{S} = (S^1, \ldots, S^l)$ is such that S^l is a vector that maps from layer-l neurons to $\{1, \ldots, k\}$. We say that the activation of a neuron is j if $w_j x + b_j$ attains the maximum among all of its functions; that is, $w_j x + b_j \geq w_{j'} x + b_{j'}$ for all $j' = 1, \ldots, j$. In the case of ties, we assume the function with lowest index is considered as its activation.

Similarly to the ReLU case, denote by $\phi_{S^l} : \mathbb{R}^{n_l \times n_{l-1} \times k} \to \mathbb{R}^{n_l \times n_{l-1}}$ the operator that selects the rows of $W_1^l, \ldots, W_k^l$ that correspond to the activations in S^l. More precisely, $\phi_{S^l}(W_1^l, \ldots, W_k^l)$ is a matrix W such that its i-th row is the i-th

row of W_j^l, where j is the neuron i's activation in S^l. This essentially applies the maxout effect on the weight matrices given an activation pattern.

Montúfar et al. (2014) provides an upper bound of $\sum_{j=0}^{n_0} \binom{k^2 n}{j}$ for the number of regions for a single rank-k maxout layer with n neurons. The reasoning is as follows. For a single maxout unit, there is one region per linear function. The boundaries between the regions are composed by pieces that are each contained in a hyperplane. Each piece is part of the boundary of at least two regions and conversely each pair of regions corresponds to at most one piece. Extending these pieces into hyperplanes cannot decrease the number of regions. Therefore, if we now consider n maxout units in a single layer, we can have at most the number of regions of an arrangement of $k^2 n$ hyperplanes. In the results below we replace k^2 by $\binom{k}{2}$, as only pairs of distinct functions need to be considered.

We need to define more precisely these $\binom{k}{2} n$ hyperplanes in order to apply a strategy similar to the one from the Section 3.1. In a single layer setting, they are given by $w_j x + b_j = w_{j'} + b_{j'}$ for each distinct pair j, j' within a neuron. In order to extend this to multiple layers, consider a $\binom{k}{2} n_l \times n_{l-1}$ matrix $\hat{W}_l$ where its rows are given by $w_j - w_{j'}$ for every distinct pair j, j' within a neuron i and for every neuron $i = 1, \ldots, n_l$. Given a region $\mathcal{S}$, we can now write the weight matrix corresponding to the hyperplanes described above: $\hat{W}_{\mathcal{S}}^l := \hat{W}^l \, \phi_{S^{l-1}}(W_1^{l-1}, \ldots, W_k^{l-1}) \cdots \phi_{S^1}(W_1^1, \ldots, W_k^1)$. In other words, the hyperplanes that extend the boundary pieces within region $\mathcal{S}$ are given by the rows of $\hat{W}_{\mathcal{S}}^l x + b = 0$ for some bias b.

A main difference between the maxout case and the ReLU case is that the maxout operator ϕ does not guarantee reductions in rank, unlike the ReLU operator σ. We show the analogous of Lemma 5 for the maxout case. However, we fully relax the rank.

Lemma 18. *The number of regions induced by the n_l neurons at layer l within a certain region $\mathcal{S}$ is at most $\sum_{j=0}^{d_l} \binom{\frac{k(k-1)}{2} n_l}{j}$, where $d_l = \min\{n_0, n_1, \ldots, n_l\}$.*

Proof. For a fixed region $\mathcal{S}$, an upper bound is given by the number of regions of the hyperplane arrangement corresponding to $\hat{W}_{\mathcal{S}}^l x + b = 0$ for some bias b. The rank of $\hat{W}_{\mathcal{S}}^l$ is upper bounded by

$$\begin{aligned}
\operatorname{rank}(\hat{W}_{\mathcal{S}}^l) &= \operatorname{rank}(\hat{W}^l \, \phi_{S^{l-1}}(W_1^{l-1}, \ldots, W_k^{l-1}) \cdots \phi_{S^1}(W_1^1, \ldots, W_k^1)) \\
&\leq \min\{\operatorname{rank}(\hat{W}^l), \operatorname{rank}(\phi_{S^{l-1}}(W_1^{l-1}, \ldots, W_k^{l-1})), \ldots, \operatorname{rank}(\phi_{S^1}(W_1^1, \ldots, W_k^1))\} \\
&\leq \min\{n_0, n_1, \ldots, n_l\}.
\end{aligned}$$

Applying Lemma 4 yields the result. □

Since we can consider the partitioning of regions independently from each other, Lemma 18 implies that the maximal number of regions of a rank-k maxout network is at most $\prod_{l=1}^{L} \sum_{j=0}^{d_l} \binom{\frac{k(k-1)}{2} n_l}{j}$ where $d_l = \min\{n_0, n_1, \ldots, n_l\}$.

□

H. Implementation of the mixed-integer formulation

In practice, the value of constant M should be chosen to be as small as possible, which also implies choosing different values on different places to make the formulation tighter and more stable numerically (Camm et al., 1990). For the constraints set (1)–(6), it suffices to choose M to be as large as either h_i^l or $\overline{h}_i^l$ can be given the bounds on the input. Hence, we can respectively replace M with H_i^l and $\bar{H}_i^l$ in the constraints involving those variables. If we are given lower and upper bounds for X, which we can use for H^0 and $\bar{H}^0$, then we can define subsequent bounds as follows:

$$H_i^l = \max\left\{0, \sum_j \max\left\{0, w_{ij}^l H_j^{l-1}\right\} + b_i^l\right\}$$

$$\overline{H}_i^l = \max\left\{0, \sum_j \max\left\{0, -w_{ij}^l H_j^{l-1}\right\} - b_i^l\right\}$$

For the constraint involving f in formulation $\mathcal{P}$, we should choose a slightly larger value than H_i^l for correctness because some neurons may never be active within the input bounds.

I. Mixed-integer representability of rectifier networks

Corollary 19. *If the input X is a polytope, then the (x, y) mapping of a rectifier DNN is mixed-integer representable.*

Proof. Immediate from the existence of a mixed-integer formulation mapping x to y, which is correct as long as the input is bounded and thus a sufficiently large M exists. □

Formulation $\mathcal{P}$ and the result above have important consequences. First, they allow us to tap into the literature of mixed-integer representability (Jeroslow, 1987) and disjunctive programming (Balas, 1979) to understand what can be modeled on rectifier networks with a finite number of neurons and layers. To the best of our knowledge, that has not been discussed before. Second, they imply that we can use mixed-integer optimization solvers to analyze the $(\mathbf{x}, \mathbf{y})$ mapping of a trained neural network. For example, Cheng et al. (2017) use another mixed-integer formulation to generate adversarial examples.

J. Counting linear regions of ReLUs with unrestricted inputs

More generally, we can represent linear regions as a disjunctive program (Balas, 1979), which consist of a union of polyhedra. Disjunctive programs are used in the integer programming literature to generate cutting planes by lift-and-project (Balas et al., 1993). In what follows, we assume that a neuron can be either active or inactive when the output lies on the activation hyperplane.

For each active neuron, we can use the following constraints to map input to output:

$$w_i^l h^{l-1} + b_i^l = h_i^l \tag{7}$$

$$h_i^l \geq 0 \tag{8}$$

For each inactive neuron, we use the following constraint:

$$w_i^l h^{l-1} + b_i^l \leq 0 \tag{9}$$

$$h_i^l = 0 \tag{10}$$

Theorem 20. *The set of linear regions of a rectifier network is a union of polyhedra.*

Proof. First, the activation set S^l for each level l defines the following mapping:

$$\bigcup_{S^l \subseteq \{1,\dots,n_l\}, l \in \{1,\dots,L+1\}} \left\{ (h^0, h^1, \dots, h^{L+1}) \mid (7)-(8) \text{ if } i \in S^l; (9)-(10) \text{ otherwise } \right\} \tag{11}$$

Consequently, we can project the variables sets $h^1, \dots, h^{L+1}$ out of each of those terms by Fourier-Motzkin elimination (Fourier, 1826), thereby yielding a polyhedron for each combination of active sets across the layers. □

Note that the result above is similar in essence to Theorem 2 of (Raghu et al., 2017).

Corollary 21. *If X is unrestricted, then the number of linear regions can be counted using $\mathcal{P}$ if M is large enough.*

Proof. To count regions, we only need one point x from each linear region. Since the number of linear regions is finite, then it suffices if M is large enough to correctly map a single point in each region. Conversely, each infeasible linear region either corresponds to empty sets of (11) or else to a polyhedron P such that $\{(h^1, \dots, h^{L+1}) \in P \mid h_i^l > 0 \; \forall l \in \{1, \dots, L+1\}, i \in S^l\}$ is empty, and neither case would yield a solution for the z-projection of $\mathcal{P}$. □

K. Mixed-integer representability of maxout units

In what follows, we assume that we are given a neuron i in level l with output h_i^l. For that neuron, we denote the vector of weights as $w_1^{li}, \dots, w_k^{li}$. Thus, the neuron output corresponds to

$$h_i^l := \max \left\{ w_1^{li} h^{l-1} + b_1, \dots, w_k^{li} h^{l-1} + b_k \right\}$$

Hence, we can connect inputs to outputs for that given neuron as follows:

$$w_j^{li} h_j^{l-1} + b_j^{li} = g_j^{li}, \qquad j = 1, \ldots, k \tag{12}$$

$$h_i^l \geq g_j^{li}, \qquad j = 1, \ldots, k \tag{13}$$

$$h_i^l \leq g_j^{li} + M(1 - z_j^{li}) \qquad j = 1, \ldots, k \tag{14}$$

$$z_j^{li} \in \{0, 1\}, \qquad j = 1, \ldots, k \tag{15}$$

$$\sum_{j=1}^{k} z_j^{li} = 1 \tag{16}$$

The formulation above generalizes that for ReLUs with some small modifications. First, we are computing the output of each term with constraint (12). The output of the neuron is lower bounded by that of each term with constraint (13). Finally, we have a binary variable z_m^{li} per term of each neuron, which denotes which term matches the output. Constraint (16) enforces that only one variable is at one per neuron, whereas constraint (14) equates the output of the neuron with the active term. Each constant M should be chosen in a way that the other terms can vary freely, hence effectively disabling the constraint when the corresponding binary variable is at zero.

L. Runtimes for counting the linear regions

Table 1 reports the median runtimes to count different configurations of networks on the experiment.

Network configuration	Runtime (s)
$1; 21; 10$	5.3×10^{-1}
$2; 20; 10$	8.0×10^{-1}
$3; 19; 10$	5.0×10^{0}
$4; 18; 10$	4.0×10^{1}
$5; 17; 10$	2.1×10^{2}
$6; 16; 10$	5.7×10^{2}
$7; 15; 10$	1.8×10^{3}
$8; 14; 10$	4.5×10^{3}
$9; 13; 10$	9.2×10^{3}
$10; 12; 10$	1.7×10^{4}
$11; 11; 10$	3.3×10^{4}
$12; 10; 10$	5.3×10^{4}
$13; 9; 10$	7.3×10^{4}
$14; 8; 10$	1.1×10^{5}
$15; 7; 10$	9.3×10^{4}
$16; 6; 10$	1.3×10^{5}
$17; 5; 10$	1.6×10^{5}
$18; 4; 10$	1.8×10^{5}
$19; 3; 10$	2.4×10^{5}
$20; 2; 10$	9.9×10^{4}
$21; 1; 10$	3.7×10^{4}

Table 1. Median runtimes for counting the trained networks for each configuration used in the experiment.