Towards Lower Bounds on the Depth of ReLU Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

We contribute to a better understanding of the class of functions that is represented by a neural network with ReLU activations and a given architecture. Using techniques from mixed-integer optimization, polyhedral theory, and tropical geometry, we provide a mathematical counterbalance to the universal approximation theorems which suggest that a single hidden layer is sufficient for learning tasks. In particular, we investigate whether the class of *exactly* representable functions *strictly* increases by adding more layers (with no restrictions on size). This problem has potential impact on algorithmic and statistical aspects because of the insight it provides into the class of functions represented by neural hypothesis classes. However, to the best of our knowledge, this question has not been investigated in the neural network literature. We also present upper bounds on the sizes of neural networks required to represent functions in these neural hypothesis classes.

1 Introduction

A core problem in machine learning or statistical pattern recognition is the estimation of an unknown data distribution with access to i.i.d. samples from the distribution. It is well-known that there is a tension between how much prior information one has about the data distribution and how many samples one needs to solve the problem with high confidence (or equivalently, how much variance one has in one's estimate). This is referred to as the *bias-variance* trade-off or the *bias-complexity* trade-off. Neural networks provide a way to turn this bias/complexity knob in a controlled manner that has been studied for decades going back to the idea of a *perceptron* by Rosenblatt [1958]. This is done by modifying the *architecture* of a neural network class of functions, which has two parameters: *depth* and *size*. As one increases these parameters, the class of functions becomes more expressive. In terms of the bias-variance trade-off, the "bias" decreases as the class of functions becomes more expressive, but the "variance" or "complexity" increases.

So-called *universal approximation theorems* [Cybenko, 1989, Hornik, 1991, Anthony and Bartlett, 1999] show that even with a single hidden layer, i.e., when the depth of the architecture is the smallest possible value, one can essentially reduce the "bias" as much as one desires, by increasing the size of the network or the number of neurons used in the neural network. Nevertheless, it can be advantageous both theoretically and empirically to increase the depth because a substantial reduction in the size can be achieved; Telgarsky [2016], Eldan and Shamir [2016], Arora et al. [2018] is a small sample of recent work in this direction. To get a better quantitative handle on these trade-offs, it is important to understand what classes of functions are exactly representable by neural networks with a certain architecture. The precise mathematical statements of universal approximation theorems show that single layer networks can *approximate* arbitrarily well any continuous function (under some additional mild hypotheses). While this suggests that single layer networks are good enough from a learning perspective, from a mathematical perspective, one can ask the question if the class of functions represented by a single layer is a *strict* subset of the class of function represented by two or

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

more hidden layers. On the question of size, one can ask for precise bounds on the size of the network of a given depth to represent a certain class of functions. We believe that a better understanding of the function classes exactly represented by different architectures will have implications not just for mathematical foundations, but also algorithmic and statistical learning aspects of neural networks. The task of searching for the "best" function in that class can only benefit from a better understanding of the nature of functions in that class. A motivating question behind the results in this paper is to understand the hierarchy of function classes exactly represented by neural networks of increasing depth.

We now introduce more precise notation and terminology to set the stage for our investigations.

Notation. We write $[n] \coloneqq \{1, 2, \ldots, n\}$ for the set of natural numbers up to n (without zero) and $[n]_0 \coloneqq [n] \cup \{0\}$ for the same set including zero. For any $n \in \mathbb{N}$, let $\sigma \colon \mathbb{R}^n \to \mathbb{R}^n$ be the component-wise *rectifier* function

$$\sigma(x) = (\max\{0, x_1\}, \max\{0, x_2\}, \ldots, \max\{0, x_n\}).$$

For any *number of hidden layers* $k \in \mathbb{N}$, a $(k+1)$*-layer feedforward neural network with rectified linear units* (ReLU NN or simply NN) is given by k affine transformations $T^{(\ell)} \colon \mathbb{R}^{n_{\ell-1}} \to \mathbb{R}^{n_\ell}$, $x \mapsto A^{(\ell)}x + b^{(\ell)}$, for $\ell \in [k]$, and a linear transformation $T^{(k+1)} \colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$, $x \mapsto A^{(k+1)}x$. It is said to *compute* or *represent* the function $f \colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_{k+1}}$ given by

$$f = T^{(k+1)} \circ \sigma \circ T^{(k)} \circ \sigma \circ \cdots \circ T^{(2)} \circ \sigma \circ T^{(1)}.$$

The matrices $A^{(\ell)} = (a_{ij}^{(\ell)})_{ij} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ are called the *weights* and the vectors $b^{(\ell)} = (b_i^{(\ell)})_i \in \mathbb{R}^{n_\ell}$ are the *biases* of the ℓ-th layer. The number $n_\ell \in \mathbb{N}$ is called the *width* of the ℓ-th layer. The maximum width of all hidden layers $\max_{\ell \in [k]} n_\ell$ is called the *width* of the NN. Further, we say that the NN has *depth* $k+1$ and *size* $\sum_{\ell=1}^{k} n_\ell$.

Often, NNs are represented as layered, directed, acyclic graphs where each dimension of each layer (including input layer $\ell = 0$ and output layer $\ell = k+1$) is one vertex, weights are arc labels, and biases are node labels. Then, the vertices are called *neurons*.

For a given input $x = x^{(0)} \in \mathbb{R}^{n_0}$, let $y^{(\ell)} \coloneqq T^{(\ell)}(x^{(\ell-1)}) \in \mathbb{R}^{n_\ell}$ be the *activation vector* and $x^{(\ell)} \coloneqq \sigma(y^\ell) \in \mathbb{R}^{n_\ell}$ the *output vector* of the ℓ-th layer. Further, let $y \coloneqq y^{(k+1)} = f(x)$ be the *output* of the NN. We also say that the i-th component of each of these vectors is the *activation* or the *output* of the i-th neuron in the ℓ-th layer.

For $k \in \mathbb{N}$, we define

$$\begin{aligned} \mathrm{ReLU}_n(k) &\coloneqq \{f \colon \mathbb{R}^n \to \mathbb{R} \mid f \text{ can be represented by a } (k+1)\text{-layer NN}\}, \\ \mathrm{CPWL}_n &\coloneqq \{f \colon \mathbb{R}^n \to \mathbb{R} \mid f \text{ is continuous and piecewise linear}\}. \end{aligned}$$

By definition, a function $f \colon \mathbb{R}^n \to \mathbb{R}$ is continuous and piecewise linear in case there is a finite set of polyhedra whose union is $\mathbb{R}^n$, and f is affine linear over each such polyhedron.

In order to analyze $\mathrm{ReLU}_n(k)$, we use another function class defined as follows. We call a function g that can be expressed as maximum of p affine terms, that is, $g(x) = \max\{\ell_1(x), \ldots, \ell_p(x)\}$ where $\ell_i \colon \mathbb{R}^n \to \mathbb{R}$ is affinely linear for $i \in [p]$, a *p-term max* function. Based on that, we define

$$\mathrm{MAX}_n(p) \coloneqq \{f \colon \mathbb{R}^n \to \mathbb{R} \mid f \text{ is a linear combination of } p\text{-term max functions}\}.$$

If the input dimension n is not important for the context, we might drop the index and write $\mathrm{ReLU}(k)$ and $\mathrm{MAX}(p)$ instead of $\mathrm{ReLU}_n(k)$ and $\mathrm{MAX}_n(p)$.

Since we are dealing a lot with polyhedra, we will use the standard notations $\operatorname{conv} A$ and $\operatorname{cone} A$ for the convex and conic hulls of a set $A \subseteq \mathbb{R}^n$. For an in-depth treatment of polyhedra and (mixed-integer) optimization, we refer to Schrijver [1986].

1.1 Our Contribution

It is not hard to see that any function expressed by a ReLU network is a continuous and piecewise linear (CPWL) function, because one is composing continuous piecewise linear functions together.

Based on a result by Wang and Sun [2005], Arora et al. [2018] show that CPWL function defined on $\mathbb{R}^n$ can be represented by a ReLU neural network with $\lceil \log_2(n+1) \rceil$ hidden layers. We wish to understand whether one can do better. We believe it is not possible to do better and we pose the following conjecture to better understand the importance of depth in neural networks.

Conjecture 1.1. *For any $n \in \mathbb{N}$, let $k^* := \lceil \log_2(n+1) \rceil$. Then it holds that*

$$\mathrm{ReLU}_n(0) \subsetneq \mathrm{ReLU}_n(1) \subsetneq \cdots \subsetneq \mathrm{ReLU}_n(k^*-1) \subsetneq \mathrm{ReLU}_n(k^*) = \mathrm{CPWL}_n \,. \tag{1}$$

Conjecture 1.1 claims that any additional layer up to k^* hidden layers strictly increases the set of representable functions. This would imply that the construction by Arora et al. [2018, Theorem 2.1] is actually depth-minimal.

Observe that in order to prove Conjecture 1.1, it suffices to find one function in $f \in \mathrm{ReLU}_n(k^*) \setminus \mathrm{ReLU}_n(k^*-1)$ with $n = 2^{k^*-1}$ for all $k^* \in \mathbb{N}$. This also implies all remaining strict inclusions $\mathrm{ReLU}_n(i-1) \subsetneq \mathrm{ReLU}_n(i)$ for $i < k^*$ since $\mathrm{ReLU}_n(i-1) = \mathrm{ReLU}_n(i)$ directly implies that $\mathrm{ReLU}_n(i-1) = \mathrm{ReLU}_n(i')$ for all $i' \geq i-1$.

In fact, there is a canonical candidate for such a function, allowing us to reformulate the conjecture as follows.

Conjecture 1.2. *For any $k \in \mathbb{N}$, $n = 2^k$, the function $f_n(x) = \max\{0, x_1, \ldots, x_n\}$ cannot be represented with k hidden layers.*

Proposition 1.3. *Conjecture 1.1 and Conjecture 1.2 are equivalent.*

Proof (Sketch). We argued above that Conjecture 1.2 implies Conjecture 1.1. The other direction can be shown using a result by [Wang and Sun, 2005] stating that any $f \in \mathrm{CPWL}_n$ can be written as linear combination of $(n+1)$-term max functions. We provide a detailed argument in Appendix A. □

It is known that Conjecture 1.2 holds for $k = 1$ [Mukherjee and Basu, 2017]. However, the conjecture remains open for $k \geq 2$. In this paper, we present the following results as partial progress towards resolving this conjecture.

In Section 2, we resolve Conjecture 1.2 for $k = 2$, under a natural assumption on the breakpoints of the function represented by any intermediate neuron. We achieve this result by leveraging techniques from mixed-integer programming to analyze the set of functions computable by certain NNs.

It is not hard to see that $\mathrm{MAX}(2^k) \subseteq \mathrm{ReLU}(k)$ for all $k \in \mathbb{N}$ [Arora et al., 2018], that is, any 2^k-term max function (and linear combinations thereof) can be expressed with k hidden layers. One might ask whether the converse is true as well, that is, whether the classes $\mathrm{MAX}(2^k)$ and $\mathrm{ReLU}(k)$ are actually equal. This would not only provide a neat characterization of $\mathrm{ReLU}(k)$, but also prove Conjecture 1.2 without any additional assumption since one can show that $\max\{0, x_1, \ldots, x_{2^k}\}$ is not contained in $\mathrm{MAX}(2^k)$.

In fact, this is true for $k = 1$, that is, a function is computable with one hidden layer if and only if it is a linear combination of 2-term max functions. However, in Section 3, we show that for $k \geq 2$, the class $\mathrm{ReLU}(k)$ is a strict superset of $\mathrm{MAX}(2^k)$. In this section, the key technical ingredient is the theory of polyhedral complexes associated with CPWL functions. This way, we provide important insights concerning the richness of the class $\mathrm{ReLU}(k)$.

So far, we have focused on understanding the smallest depth needed to express CPWL functions using neural networks with ReLU activations. In Section 4, we complement these results by upper bounds on the sizes of the networks needed for expressing arbitrary CPWL functions. In particular, Theorem 4.4 shows that any continuous piecewise linear function with p linear/affine pieces on $\mathbb{R}^n$ can be expressed by a network with depth at most $O(\log n)$ and width at most $p^{O(n^2)}$. We arrive at this result by introducing a novel application of recently established interactions between neural networks and tropical geometry.

1.2 Related Work

Depth versus size. Soon after the original universal approximation theorems Cybenko [1989], Hornik [1991], concrete bounds were obtained on the number of neurons needed in the hidden layer to

achieve a certain level of accuracy. The literature on this is vast and we refer to a small representative sample here Barron [1993, 1994], Mhaskar [1993], Pinkus [1999], Mhaskar [1996], Mhaskar and Micchelli [1995]. More recently, work has focused on how deeper networks can have exponentially or super exponentially smaller size compared to shallower networks Telgarsky [2016], Eldan and Shamir [2016], Arora et al. [2018]. See also Gribonval et al. [2021] for another perspective on the relationship between expressivity and architecture, and the references therein. We reiterate that the list of references above is far from complete.

Mixed-integer optimization and machine learning. Over the past decade, a growing body of work has emerged that explores the interplay between mixed-integer optimization and machine learning. On the one hand, researchers have attempted to improve mixed-integer optimization algorithms by exploiting novel techniques from machine learning Bonami et al. [2018], Gasse et al. [2019], He et al. [2014], Khalil et al. [2016, 2017], Kruber et al. [2017], Lodi and Zarpellon [2017], Alvarez et al. [2017]; see also Bengio et al. [2020] for a recent survey. On the flip side, mixed-integer optimization techniques have been used to analyze function classes represented by neural networks Serra et al. [2018], Anderson et al. [2020], Fischetti and Jo [2017], Serra and Ramalingam [2020], Serra et al. [2020]. In Section 2 below, we show another new use of mixed-integer optimization tools for understanding function classes represented by neural networks.

Design of training algorithms. We believe that a better understanding of the function classes represented exactly by a neural architecture also has benefits in terms of understanding the complexity of the training problem. For instance, in Arora et al. [2018], an understanding of single layer ReLU networks enables the design of a globally optimal algorithm for solving the empirical risk minimization (ERM) problem, that runs in polynomial time in the number of data points in fixed dimension. See also Goel et al. [2017, 2018], Goel and Klivans [2019], Dey et al. [2020], Boob et al. [2020], Goel et al. [2021], Froese et al. [2021] for a similar lines of work.

Neural Networks and Tropical Geometry. A recent stream of research involves the interplay between neural networks and tropical geometry. The piecewise linear functions computed by neural networks can be seen as (tropical quotients of) tropical polynomials. Linear regions of these functions correspond to vertices of so-called *Newton polytopes* associated with these tropical polynomials. Applications of this correspondence include bounding the number of linear regions of a neural network [Zhang et al., 2018, Charisopoulos and Maragos, 2018, Montúfar et al., 2021] and understanding decision boundaries [Alfarra et al., 2020]. In Section 4 we present a novel application of tropical concepts to understand neural networks. We refer to Maragos et al. [2021] for a recent survey of connections between machine learning and tropical geometry, as well as to the textbooks by Maclagan and Sturmfels [2015] and Joswig [2022] for in-depth introductions to tropical geometry and tropical combinatorics.

2 Conditional Lower Bounds on Depth via Mixed-Integer Programming

In this section, we provide a computer-aided proof that, under a natural, yet unproven assumption, the function $f(x) \coloneqq \max\{0, x_1, x_2, x_3, x_4\}$ cannot be represented by a 3-layer NN. For easier notation, we write $x_0 \coloneqq 0$.

We first prove that we may restrict ourselves to NNs without biases. This holds true independent of our assumption, which we introduce afterwards.

Definition 2.1. A function $g\colon \mathbb{R}^n \to \mathbb{R}^m$ is called *positively homogeneous* if $g(\lambda x) = \lambda g(x)$ for all $\lambda \geq 0$.

Definition 2.2. For an NN given by affine transformations $T^{(\ell)}(x) = A^{(\ell)}x + b^{(\ell)}$, we defined the corresponding *homogenized NN* to be the NN given by $\tilde{T}^{(\ell)}(x) = A^{(\ell)}x$ with all biases set to zero.

Proposition 2.3. *If an NN computes a positively homogeneous function, then the corresponding homogenized NN computes the same function.*

Proof. Let $g\colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_{k+1}}$ be the function computed by the original NN and $\tilde{g}$ the one computed by the homogenized NN. Further, for any $0 \leq \ell \leq k$, let $g^{(\ell)} = T^{(\ell+1)} \circ \sigma \circ T^{(\ell)} \circ \cdots \circ T^{(2)} \circ \sigma \circ T^{(1)}$ be the function computed by the sub-NN consisting of the first $(\ell+1)$-layers and let $\tilde{g}^{(\ell)}$ be the

function computed by the corresponding homogenized sub-NN. We first show by induction on ℓ that the norm of $\|g^{(\ell)}(x) - \tilde{g}^{(\ell)}(x)\|$ is bounded by a global constant that only depends on the parameters of the NN but not on x.

For $\ell = 0$, we obviously have $\|g^{(0)}(x) - \tilde{g}^{(0)}(x)\| = \|b^{(1)}\| =: C_0$, settling the induction base. For the induction step, let $\ell \geq 1$ and assume that $\|g^{(\ell-1)}(x) - \tilde{g}^{(\ell-1)}(x)\| \leq C_{\ell-1}$, where $C_{\ell-1}$ only depends on the parameters of the NN. Since component-wise ReLU activation has Lipschitz constant 1, this implies $\|(\sigma \circ g^{(\ell-1)})(x) - (\sigma \circ \tilde{g}^{(\ell-1)})(x)\| \leq C_{\ell-1}$. Using any matrix norm that is compatible with the Euclidean vector norm, we obtain:

$$\begin{aligned} \|g^{(\ell)}(x) - \tilde{g}^{(\ell)}(x)\| &= \|b^{(\ell+1)} + A^{(\ell+1)}((\sigma \circ g^{(\ell-1)})(x) - (\sigma \circ \tilde{g}^{(\ell-1)})(x))\| \\ &\leq \|b^{(\ell+1)}\| + \|A^{(\ell+1)}\| \cdot C_{\ell-1} =: C_\ell \end{aligned}$$

Since the right-hand side only depends on NN parameters, the induction is completed.

Finally, we show that $g = \tilde{g}$. For the sake of contradiction, suppose that there is an $x \in \mathbb{R}^{n_0}$ with $\|g(x) - \tilde{g}(x)\| = \delta > 0$. Let $x' := \frac{C_k+1}{\delta} x$; then, by positive homogeneity, it follows that $\|g(x') - \tilde{g}(x')\| = C_k + 1 > C_k$, contradicting the property shown above. Thus, $g = \tilde{g}$ follows. □

Since f is positively homogeneous, Proposition 2.3 implies that, if there is a 3-layer NN computing f, then there also is one that has no biases. Therefore, in the remainder of this section, we only consider NNs without biases and assume implicitly that all considered CPWL functions are positively homogeneous. In particular, any piece of such a CPWL function is linear and not only affine linear.

Observe that, for function f, the only points of non-differentiability (a.k.a. *breakpoints*) are at places where at least two of the five numbers $x_0 = 0$, x_1, x_2, x_3, and x_4 are equal. Hence, if some neuron of an NN computing f introduces breakpoints at other places, these breakpoints must be canceled out by other neurons. Therefore, it is a natural assumption that such breakpoints are not introduced at all in the first place.

To make this assumption formal, let $H_{ij} = \{x \in \mathbb{R}^4 \mid x_i = x_j\}$, for $0 \leq i < j \leq 4$, be ten hyperplanes in $\mathbb{R}^4$ and $H = \bigcup_{0 \leq i < j \leq 4} H_{ij}$ be the corresponding hyperplane arrangement. The *regions* or *cells* of H are defined to be the closures of the connected components of $\mathbb{R}^4 \setminus H$. It is easy to see that these regions are in one-to-one correspondence to the $5! = 120$ possible orderings of the five numbers $x_0 = 0$, x_1, x_2, x_3, and x_4. More precisely, for a permutation π of the five indices $[4]_0 = \{0, 1, 2, 3, 4\}$, the corresponding region is the polyhedron

$$C_\pi := \{x \in \mathbb{R}^4 \mid x_{\pi(0)} \leq x_{\pi(1)} \leq x_{\pi(2)} \leq x_{\pi(3)} \leq x_{\pi(4)}\}.$$

We say that a (positively homogeneous) CPWL function g is *H-conforming*, if it is linear within any of these regions of H, that is, if it only has breakpoints where the relative ordering of the five values $x_0 = 0$, x_1, x_2, x_3, x_4 changes. Moreover, an NN is said to be H-conforming if the output of each neuron contained in the NN is H-conforming. Equivalently, this is the case if and only if all intermediate functions $\sigma \circ T^{(\ell)} \circ \sigma \circ T^{(\ell-1)} \circ \cdots \circ \sigma \circ T^{(1)}$, $\ell \in [k]$, are H-conforming. Now our assumption can be formally phrased as follows.

Assumption 2.4. *If there exists a 3-layer NN computing $f(x) = \max\{0, x_1, x_2, x_3, x_4\}$, then there also exists one that is H-conforming.*

We use mixed-integer programming to prove the following theorem. It is worth to note that, to the best of our knowledge, no CPWL function is known for which the non-existence of a 3-layer NN can be proven without additional assumption.

Theorem 2.5. *Under Assumption 2.4, there does not exist a 3-layer NN that computes the function $f(x) = \max\{0, x_1, x_2, x_3, x_4\}$.*

Proof (Outline). We first study some geometric properties of the hyperplane arrangement H. This will show that each of the 120 cells of H is a simplicial polyhedral cone spanned by 4 extreme rays. In total, there are 30 such rays (because rays are used multiple times to span different cones). This implies that the set of H-conforming functions of type $\mathbb{R}^4 \to \mathbb{R}$ is a 30-dimensional vector space and each function is uniquely determined by its values on the 30 rays. We then use linear algebra to show that the space of functions generated by H-conforming two-layer NNs is a 14-dimensional

subspace. Moreover, with two hidden layers, at least 29 of the 30 dimensions can be generated and that f is not contained in this 29-dimensional subspace. So, it remains the question whether the 14 dimensions producible with the first hidden layer can be combined in such a way that after applying a ReLU activation in the second hidden layer, we do not end up within the 29-dimensional subspace. We model this question as a mixed-interger program (MIP). Solving the MIP yields that we always end up within the 29-dimensional subspace, implying that f cannot be represented by a 3-layer NN. This provides a computational proof of Theorem 2.5. All details can be found in Appendix B. □

3 Going Beyond 2^k-Term Max Functions with k Layers

In this section we prove the following result:

Theorem 3.1. *For any $k \geq 2$, the class $\mathrm{ReLU}(k)$ is a strict superset of $\mathrm{MAX}(2^k)$.*

In order to prove this theorem, we provide a specific function that is in $\mathrm{ReLU}(k) \setminus \mathrm{MAX}(2^k)$ for any number of hidden layers $k \geq 2$. The challenging part is to show that the function is in fact not contained in $\mathrm{MAX}(2^k)$.

Proposition 3.2. *For any $n \geq 3$, the function*

$$f\colon \mathbb{R}^n \to \mathbb{R}, \quad f(x) = \max\{0, x_1, x_2, \ldots, x_{n-3}, \max\{x_{n-2}, x_{n-1}\} + \max\{0, x_n\}\} \tag{2}$$

cannot be written as a linear combination of n-term max functions.

The above proposition means that it is not possible to write $f(x)$ in the form

$$f(x) = \sum_{i=1}^{p} \lambda_i \max\{\ell_{i1}(x), \ldots, \ell_{in}(x)\}$$

where $p \in \mathbb{N}$, $\lambda_1, \ldots, \lambda_p \in \mathbb{R}$ and $\ell_{ij} : \mathbb{R}^n \to \mathbb{R}$ is an affine linear function for every $i \in [p]$ and $j \in [n]$. (Note that max functions with less than n terms are also allowed, as some functions ℓ_{ij} may coincide.)

Before we prove Proposition 3.2, we show that it implies Theorem 3.1.

Proof of Theorem 3.1. For $k \geq 2$, let $n \coloneqq 2^k$. By Proposition 3.2, function f defined in (2) is not contained in $\mathrm{MAX}(2^k)$. It remains to show that it can be represented using a ReLU NN with k hidden layers. To see this, first observe that any of the $n/2 = 2^{k-1}$ terms $\max\{0, x_1\}$, $\max\{x_{2i}, x_{2i+1}\}$ for $i \in [n/2 - 2]$, and $\max\{x_{n-2}, x_{n-1}\} + \max\{0, x_n\}$ can be expressed by a one-hidden-layer NN since all these are (linear combinations of) 2-term max functions. Since f is the maximum of these 2^{k-1} terms, and since the maximum of 2^{k-1} numbers can be computed with $k-1$ hidden layers [Arora et al., 2018], this implies that f is in $\mathrm{ReLU}(k)$. □

In order to prove Proposition 3.2, we need the concept of polyhedral complexes. A *polyhedral complex* $\mathcal{P}$ is a set of polyhedra such that each face of a polyhedron in $\mathcal{P}$ is also in $\mathcal{P}$, and for two polyhedra $P, Q \in \mathcal{P}$, their intersection $P \cap Q$ is a common face of P and Q (possibly the empty face). Given a polyhedral complex $\mathcal{P}$ in $\mathbb{R}^n$ and an integer $m \in [n]$, we let $\mathcal{P}^m$ denote the collection of all m-dimensional polyhedra in $\mathcal{P}$.

For a convex CPWL function f, we define its *underlying polyhedral complex* as follows: it is the unique polyhedral complex covering $\mathbb{R}^n$ (i.e., each point in $\mathbb{R}^n$ belongs to some polyhedron in $\mathcal{P}$) whose n-dimensional polyhedra coincide with the domains of the (maximal) affine pieces of f. In particular, f is affinely linear within each $P \in \mathcal{P}$, but not within any strict superset of a polyhedron in $\mathcal{P}^n$.

Exploiting properties of polyhedral complexes associated with CPWL functions, we prove the following proposition in Appendix C.

Proposition 3.3. *Let $f_0\colon \mathbb{R}^n \to \mathbb{R}$ be a convex CPWL function and let $\mathcal{P}_0$ be the underlying polyhedral complex. If there exists a hyperplane $H \subseteq \mathbb{R}^n$ such that the set*

$$T \coloneqq \bigcup \left\{ F \in \mathcal{P}_0^{n-1} \;\middle|\; F \subseteq H \right\}$$

is nonempty and contains no line, then f_0 cannot be expressed as a linear combination of n-term maxima of affine linear functions.

This allows us to prove Proposition 3.2.

Proof of Proposition 3.2. Observe that f (defined in (2)) has the alternate representation

$$f(x) = \max\{0,\, x_1,\, x_2,\, \ldots,\, x_{n-3},\, x_{n-2},\, x_{n-1},\, x_{n-2} + x_n,\, x_{n-1} + x_n\}$$

as a maximum of $n+2$ terms. Let $\mathcal{P}$ be its underlying polyhedral complex. Let the hyperplane H be defined by $x_1 = 0$.

Observe that any facet in $\mathcal{P}^{n-1}$ is a polyhedron defined by two of the $n+2$ terms that are equal and at least as large as each of the remaining n terms. Hence, the only facet that could possibly be contained in H is

$$F \coloneqq \{x \in \mathbb{R}^n \mid x_1 = 0 \geq x_2,\, \ldots,\, x_{n-3},\, x_{n-2},\, x_{n-1},\, x_{n-2} + x_n,\, x_{n-1} + x_n\}.$$

Note that F is indeed an $(n-1)$-dimensional facet in in $\mathcal{P}^{n-1}$, because, for example, the full neighborhood of $(0, -1, \ldots, -1) \in \mathbb{R}^n$ intersected with H is contained in F.

Finally, we need to show that F is pointed, that is, it contains no line. A well-known fact from polyhedral theory says if there is any line in F with direction $d \in \mathbb{R}^n \setminus \{0\}$, then d must satisfy the defining inequalities with equality. However, only the zero vector does this. Hence, F cannot contain a line.

Therefore, when applying Proposition 3.3 to f with underlying polyhedral complex $\mathcal{P}$ and hyperplane H, we have $T = F$, which is nonempty and contains no line. Hence, f cannot be written as linear combination of n-term maxima. □

4 A Width Bound for NNs with Small Depth

While the arguments in Arora et al. [2018] show that $\mathrm{CPWL}_n = \mathrm{ReLU}_n(\lceil \log_2(n+1) \rceil)$, they do not provide any bound on the width of the NN required to represent any particular continuous piecewise linear function. The purpose of this section is to prove that for fixed dimension n, the required width for exact, depth-minimal representation of a CPWL function can be polynomially bounded in the number p of affine pieces; in particular $p^{O(n^2)}$. This is closely related to works that bound the number of linear pieces of an NN as a function of the size [Montúfar et al., 2014, Pascanu et al., 2014, Raghu et al., 2017, Montúfar et al., 2021]. It can also be seen as a counterpart, in the context of exact representations, to quantitative universal approximation theorems that bound the number of neurons required to achieve a certain approximation guarantee; see, e.g., Barron [1993, 1994], Mhaskar [1993], Pinkus [1999], Mhaskar [1996], Mhaskar and Micchelli [1995].

4.1 The Convex Case

We first derive our result for the case of convex CPWL functions and then use this to also prove the general nonconvex case. Our width bound is a consequence of the following theorem about convex CPWL functions, for which we are going to provide a geometric proof later.

Theorem 4.1. *Let $f(x) = \max\{a_i^T x + b_i \mid i \in [p]\}$ be a convex CPWL function defined on $\mathbb{R}^n$. Then f can be written as*

$$f(x) = \sum_{\substack{S \subseteq [p],\\ |S| \leq n+1}} c_S \max\{a_i^T x + b_i \mid i \in S\}$$

with coefficients $c_S \in \mathbb{Z}$, for $S \subseteq [p]$, $|S| \leq n+1$.

For the convex case, this yields a stronger version of the theorem by Wang and Sun [2005] stating that any (not necessarily convex) CPWL function can be written as a linear combination of $(n+1)$-term maxima. Theorem 4.1 is stronger in the sense that it guarantees that all pieces of the $(n+1)$-term maxima must be pieces of the original function, making it possible to bound the total number of these $(n+1)$-term maxima and, therefore, the size of an NN representing f.

Theorem 4.2. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a convex CPWL function with p affine pieces. Then f can be represented by a ReLU NN with depth $\lceil \log_2(n+1) \rceil + 1$ and width $\mathcal{O}(p^{n+1})$.*

Proof. Since the number of possible subsets $S \subseteq [p]$ with $|S| \leq n+1$ is bounded by p^{n+1}, the theorem follows by Theorem 4.1 and the construction from Arora et al. [2018, Theorem 2.1]. $\square$

Before we present the proof of Theorem 4.1, we show how we can generalize its consequences to the nonconvex case.

4.2 The General (Nonconvex) Case

It is a well-known fact that every CPWL function can be expressed as a difference of two convex CPWL functions, see, e.g., Wang [2004, Theorem 1]. This allows us to derive the general case from the convex case. What we need, however, is to bound the number of affine pieces of the two convex CPWL functions in terms of the number of pieces of the original function. Therefore, we consider a specific decomposition for which such bounds can easily be achieved.

Proposition 4.3. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a CPWL function with p affine pieces. Then, f can be written as $f = g - h$ where both g and h are convex CPWL functions with at most p^{2n+1} pieces.*

Proof. Suppose the p affine pieces of f are given by $x \mapsto a_i^T x + b_i$, $i \in [p]$. Define the function $h(x) \coloneqq \sum_{1 \leq i < j \leq p} \max\{a_i^T x + b_i, a_j^T x + b_j\}$ and let $g \coloneqq f + h$. Then, obviously, $f = g - h$. It remains to show that both g and h are convex CPWL functions with at most p^{2n+1} pieces.

The convexity of h is clear by definition. Consider the $\binom{p}{2} = \frac{p(p-1)}{2} < p^2$ hyperplanes given by $a_i^T x + b_i = a_j^T x + b_j$, $1 \leq i < j \leq p$. They divide $\mathbb{R}^n$ into at most $\binom{p^2}{n} + \binom{p^2}{n-1} + \cdots + \binom{p^2}{0} \leq p^{2n}$ regions (compare Edelsbrunner [1987, Theorem 1.3]) in each of which h is affine. In particular, h has at most $p^{2n} \leq p^{2n+1}$ pieces.

Next, we show that $g = f + h$ is convex. Intuitively, this holds because each possible breaking hyperplane of f is made convex by adding h. To make this formal, note that by the definition of convexity, it suffices to show that g is convex along each affine line. For this purpose, consider an arbitrary line $x(t) = ta + b$, $t \in \mathbb{R}$, given by $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Let $\tilde{f}(t) \coloneqq f(x(t))$, $\tilde{g}(t) \coloneqq g(x(t))$, and $\tilde{h}(t) \coloneqq h(x(t))$. We need to show that $\tilde{g}\colon \mathbb{R} \to \mathbb{R}$ is a convex function. Observe that $\tilde{f}$, $\tilde{g}$, and $\tilde{h}$ are clearly one-dimensional CPWL functions with the property $\tilde{g} = \tilde{f} + \tilde{h}$. Hence, it suffices to show that $\tilde{g}$ is convex locally around each of its breakpoints. Let $t \in \mathbb{R}$ be an arbitrary breakpoint of $\tilde{g}$. If $\tilde{f}$ is already convex locally around t, then the same holds for $\tilde{g}$ as well since $\tilde{h}$ inherits convexity from h. Now suppose that t is a nonconvex breakpoint of $\tilde{f}$. Then there exist two distinct pieces of f, indexed by $i, j \in [p]$ with $i \neq j$, such that $\tilde{f}(t') = \min\{a_i^T x(t') + b_i, a_j^T x(t') + b_j\}$ for all t' sufficiently close to t. By construction, $\tilde{h}(t')$ contains the summand $\max\{a_i^T x(t') + b_i, a_j^T x(t') + b_j\}$. Thus, adding this summand to $\tilde{f}$ linearizes the nonconvex breakpoint of $\tilde{f}$, while adding all the other summands preserves convexity. In total, $\tilde{g}$ is convex locally around t, which finishes the proof that g is a convex function.

Finally, observe that pieces of $g = f + h$ are always intersections of pieces of f and h, for which we have only $p \cdot p^{2n} = p^{2n+1}$ possibilities. $\square$

Having this, we may conclude the following.

Theorem 4.4. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a CPWL function with p affine pieces. Then f can be represented by a ReLU NN with depth $\lceil \log_2(n+1) \rceil + 1$ and width $\mathcal{O}(p^{2n^2+3n+1})$.*

Proof. Consider the decomposition $f = g - h$ from Proposition 4.3. Using Theorem 4.2, we obtain that both g and h can be represented with the required depth $\lceil \log_2(n+1) \rceil + 1$ and with width $\mathcal{O}((p^{2n+1})^{n+1}) = \mathcal{O}(p^{2n^2+3n+1})$. Thus, the same holds for f. $\square$

4.3 Extended Newton Polyhedra of Convex CPWL Functions

For our proof of Theorem 4.1, we use a correspondence of convex CPWL functions with certain polyhedra, which are known as (extended) Newton polyhedra in tropical geometry [Maclagan and

Sturmfels, 2015]. These relations between tropical geometry and neural networks have previously been applied to investigate expressivity of NNs; compare our references in the introduction.

In order to formalize this correspondence, let $\mathrm{CCPWL}_n \subseteq \mathrm{CPWL}_n$ be the set of convex CPWL functions of type $\mathbb{R}^n \to \mathbb{R}$. For $f(x) = \max\{a_i^T x + b_i \mid i \in [p]\}$ in CCPWL_n, we define its so-called *extended Newton polyhedron* to be

$$\mathcal{N}(f) \coloneqq \mathrm{conv}(\{(a_i, b_i) \in \mathbb{R}^n \times \mathbb{R} \mid i \in [p]\}) + \mathrm{cone}(\{-e_{n+1}\}) \subseteq \mathbb{R}^{n+1}.$$

We denote the set of all possible extended Newton polyhedra as $\mathrm{Newt}_n \subseteq \mathbb{R}^{n+1}$. That is, Newt_n is the set of (unbounded) polyhedra in $\mathbb{R}^{n+1}$ that emerge from a polytope by adding the negative of the $(n+1)$-st unit vector $-e_{n+1}$ as an extreme ray. Hence, a set $P \subseteq \mathbb{R}^{n+1}$ is an element of Newt_n if and only if P can be written as $P = \mathrm{conv}(\{(a_i, b_i) \in \mathbb{R}^n \times \mathbb{R} \mid i \in [p]\}) + \mathrm{cone}(\{-e_{n+1}\})$. Conversely, for a polyhedron $P \in \mathrm{Newt}_n$ of this form, let $\mathcal{F}(P) \in \mathrm{CCPWL}_n$ be the function defined by $\mathcal{F}(P)(x) = \max\{a_i^T x + b_i \mid i \in [p]\}$.

There is an intuitive way of thinking about the extended Newton polyhedron P of a convex CPWL function f: it consists of all hyperplane coefficients $(a, b) \in \mathbb{R}^n \times \mathbb{R}$ such that $a^T x + b \leq f(x)$ for all $x \in \mathbb{R}^n$. This also explains why we add the extreme ray $-e_{n+1}$: decreasing b obviously maintains the property of $a^T x + b$ being a lower bound on the function f.

We need the notion of the *Minkowski sum* of two polyhedra P and Q: it is given as the set $P + Q = \{p + q \mid p \in P, q \in Q\}$.

In fact, there is a one-to-one correspondence between elements of CCPWL_n and Newt_n, which is nicely compatible with some (functional and polyhedral) operations. This correspondence has been studied before in the tropical geometry [Maclagan and Sturmfels, 2015, Joswig, 2022], convex geometry[1] [Hiriart-Urruty and Lemaréchal, 1993], as well as neural network literature [Zhang et al., 2018, Charisopoulos and Maragos, 2018, Alfarra et al., 2020, Montúfar et al., 2021]. We summarize the key findings of this correspondence relevant to our work in the following proposition:

Proposition 4.5. *Let $n \in \mathbb{N}$ and $f_1, f_2 \in \mathrm{CCPWL}_n$. Then it holds that*

(i) the functions $\mathcal{N}\colon \mathrm{CCPWL}_n \to \mathrm{Newt}_n$ and $\mathcal{F}\colon \mathrm{Newt}_n \to \mathrm{CCPWL}_n$ are well-defined, that is, their output is independent from the representation of the input by pieces or vertices, respectively,

(ii) $\mathcal{N}$ and $\mathcal{F}$ are bijections and inverse to each other,

(iii) $\mathcal{N}(\max\{f_1, f_2\}) = \mathrm{conv}(\mathcal{N}(f_1), \mathcal{N}(f_2)) \coloneqq \mathrm{conv}(\mathcal{N}(f_1) \cup \mathcal{N}(f_2))$,

(iv) $\mathcal{N}(f_1 + f_2) = \mathcal{N}(f_1) + \mathcal{N}(f_2)$, where the $+$ on the right-hand side is Minkowski addition.

An algebraic way of phrasing this proposition is as follows: $\mathcal{N}$ and $\mathcal{F}$ are isomorphisms between the semirings $(\mathrm{CCPWL}_n, \max, +)$ and $(\mathrm{Newt}_n, \mathrm{conv}, +)$.

4.4 Outline of the Proof of Theorem 4.1

We prove Theorem 4.1 in full detail in Appendix D. The rough idea is as follows.

Suppose we have a p-term max function f with $p \geq n + 2$. By Proposition 4.5, f corresponds to a polyhedron $P \in \mathrm{Newt}_n$ with at least $n + 2$ vertices. Applying a classical result from discrete geometry known as *Radon's theorem* allows us to carefully decompose P into a "signed"[2] Minkowski sum of polyhedra in Newt_n whose vertices are subsets of at most $p - 1$ out of the p vertices of P. Translating this back into the world of CPWL functions by Proposition 4.5 yields that f can be written as linear combination of p'-term maxima with $p' < p$, where each of them involves a subset of the p affine terms of f. We can then obtain Theorem 4.1 by iterating until every occurring maximum expression involves at most $n + 1$ terms.

[1] $\mathcal{N}(f)$ is the negative of the epigraph of the convex conjugate of f.

[2] Some polyhedra may occur with "negative" coefficents in that sum, meaning that they are actually added to P instead of the other polyhedra. The corresponding CPWL functions will then have negative coefficients in the linear combination representing f.

References

M. Alfarra, A. Bibi, H. Hammoud, M. Gaafar, and B. Ghanem. On the decision boundaries of deep neural networks: A tropical geometry perspective. *arXiv preprint arXiv:2002.08838*, 2020.

A. M. Alvarez, Q. Louveaux, and L. Wehenkel. A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, 29(1):185–195, 2017.

R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, pages 1–37, 2020.

M. Anthony and P. L. Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 1999.

R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

A. R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

P. Bonami, A. Lodi, and G. Zarpellon. Learning a classification of mixed-integer quadratic programming problems. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 595–604. Springer, 2018.

D. Boob, S. S. Dey, and G. Lan. Complexity of training relu neural network. *Discrete Optimization*, page 100620, 2020.

V. Charisopoulos and P. Maragos. A tropical approach to neural networks with piecewise linear activations. *arXiv preprint arXiv:1805.08749*, 2018.

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

S. S. Dey, G. Wang, and Y. Xie. Approximation algorithms for training one-node relu neural networks. *IEEE Transactions on Signal Processing*, 68:6696–6706, 2020.

H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Science & Business Media, 1987.

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

M. Fischetti and J. Jo. Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *arXiv preprint arXiv:1712.06174*, 2017.

V. Froese, C. Hertrich, and R. Niedermeier. The computational complexity of ReLU network training parameterized by data dimensionality. *arXiv preprint arXiv:2105.08675*, 2021.

M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. *arXiv preprint arXiv:1906.01629*, 2019.

S. Goel and A. R. Klivans. Learning neural networks with two nonlinear layers in polynomial time. In *Conference on Learning Theory*, pages 1470–1499. PMLR, 2019.

S. Goel, V. Kanade, A. Klivans, and J. Thaler. Reliably learning the relu in polynomial time. In *Conference on Learning Theory*, pages 1004–1042. PMLR, 2017.

S. Goel, A. Klivans, and R. Meka. Learning one convolutional layer with overlapping patches. In *International Conference on Machine Learning*, pages 1783–1791. PMLR, 2018.

S. Goel, A. R. Klivans, P. Manurangsi, and D. Reichman. Tight hardness results for training depth-2 ReLU networks. In *12th Innovations in Theoretical Computer Science Conference (ITCS '21)*, volume 185 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

R. Gribonval, G. Kutyniok, M. Nielsen, and F. Voigtlaender. Approximation spaces of deep neural networks. *Constructive Approximation*, pages 1–109, 2021.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2021. URL `http://www.gurobi.com`.

H. He, H. Daume III, and J. M. Eisner. Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27:3293–3301, 2014.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*, volume 306 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1993. ISBN 3-540-56852-2.

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

M. Joswig. *Essentials of tropical combinatorics*. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2022. URL `https://page.math.tu-berlin.de/~joswig/etc/index.html`. To appear.

E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

E. B. Khalil, B. Dilkina, G. L. Nemhauser, S. Ahmed, and Y. Shao. Learning to run heuristics in tree search. In *IJCAI*, pages 659–666, 2017.

M. Kruber, M. E. Lübbecke, and A. Parmentier. Learning when to use a decomposition. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 202–210. Springer, 2017.

A. Lodi and G. Zarpellon. On learning and branching: a survey. *Top*, 25(2):207–236, 2017.

D. Maclagan and B. Sturmfels. *Introduction to tropical geometry*, volume 161 of *Graduate Studies in Mathematics*. American Mathematical Soc., 2015.

P. Maragos, V. Charisopoulos, and E. Theodosis. Tropical geometry and machine learning. *Proceedings of the IEEE*, 109(5):728–755, 2021.

H. Mhaskar. Approximation of real functions using neural networks. In *Proc. Intl. Conf. Comp. Math., New Delhi, India, World Scientific Press*, pages 267–278. World Scientific, 1993.

H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8(1):164–177, 1996.

H. N. Mhaskar and C. A. Micchelli. Degree of approximation by neural and translation networks with a single hidden layer. *Advances in applied mathematics*, 16(2):151–183, 1995.

G. Montúfar, Y. Ren, and L. Zhang. Sharp bounds for the number of regions of maxout networks and vertices of minkowski sums. *arXiv preprint arXiv:2104.08135*, 2021.

G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. 2014.

A. Mukherjee and A. Basu. Lower bounds over boolean inputs for deep neural networks with relu gates. *arXiv preprint arXiv:1711.03073*, 2017.

R. Pascanu, G. Montúfar, and Y. Bengio. On the number of inference regions of deep feed forward networks with piece-wise linear activations. In *International Conference on Learning Representations*, 2014.

A. Pinkus. Approximation theory of the mlp model. *Acta Numerica 1999: Volume 8*, 8:143–195, 1999.

M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, pages 2847–2854, 2017.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, 1986.

T. Serra and S. Ramalingam. Empirical bounds on linear regions of deep rectifier networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5628–5635, 2020.

T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR, 2018.

T. Serra, A. Kumar, and S. Ramalingam. Lossless compression of deep neural networks. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 417–430. Springer, 2020.

M. Telgarsky. benefits of depth in neural networks. In *29th Annual Conference on Learning Theory*, pages 1517–1539, 2016.

The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.0)*, 2020. `https://www.sagemath.org`.

S. Wang. General constructive representations for continuous piecewise-linear functions. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(9):1889–1896, 2004.

S. Wang and X. Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

L. Zhang, G. Naitzat, and L.-H. Lim. Tropical geometry of deep neural networks. In *International Conference on Machine Learning*, pages 5819–5827, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Partially in the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] No training involved, just a MIP for which we provide full details.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] No randomness involved.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Towards the end of Appendix B.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] SageMath, Parma Polyhedral Library, and Gurobi, see Appendix B for references.

(b) Did you mention the license of the assets? [Yes] Towards the end of Appendix B.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide a SageMath script, as well as a `.mps` file in the supplementary material to execute the MIP described in Section 2 and Appendix B.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No data used.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No data used.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]