Understanding Deep Neural Networks with Rectified Linear Units

Raman Arora[*] Amitabh Basu[†] Poorya Mianjy[‡] Anirbit Mukherjee[§]

Abstract

In this paper we investigate the family of functions representable by deep neural networks (DNN) with rectified linear units (ReLU). We give the first-ever polynomial time (in the size of data) algorithm to train to *global optimality* a ReLU DNN with one hidden layer, assuming the input dimension and number of nodes of the network as fixed constants.

We also improve on the known lower bounds on size (from exponential to super exponential) for approximating a ReLU deep net function by a shallower ReLU net. Our gap theorems hold for smoothly parametrized families of "hard" functions, contrary to countable, discrete families known in the literature. An example consequence of our gap theorems is the following: for every natural number k there exists a function representable by a ReLU DNN with k^2 hidden layers and total size k^3, such that any ReLU DNN with at most k hidden layers will require at least $\frac{1}{2}k^{k+1} - 1$ total nodes.

Finally, we construct a family of $\mathbb{R}^n \to \mathbb{R}$ piecewise linear functions for $n \geq 2$ (also smoothly parameterized), whose number of affine pieces scales exponentially with the dimension n at any fixed size and depth. To the best of our knowledge, such a construction with exponential dependence on n has not been achieved by previous families of "hard" functions in the neural nets literature. This construction utilizes the theory of zonotopes from polyhedral theory.

1 Introduction

Deep neural networks (DNNs) provide an excellent family of hypotheses for machine learning tasks such as classification. Neural networks with a single hidden layer of finite size can represent any continuous function on a compact subset of $\mathbb{R}^n$ arbitrary well. The universal approximation result was first given by Cybenko in 1989 for sigmoidal activation function [6], and later generalized by Hornik to an arbitrary bounded and nonconstant activation function [16]. Furthermore, neural networks have finite VC dimension (depending polynomially on the number of edges in the network), and therefore, are PAC (probably approximately correct) learnable using a sample of size that is polynomial in the size of the networks [2]. However, neural networks based methods were shown

[*]Department of Computer Science, Johns Hopkins University, Email: `arora@cs.jhu.edu`

[†]Department of Applied Mathematics and Statistics, Johns Hopkins University, Email: `basu.amitabh@jhu.edu`

[‡]Department of Computer Science, Johns Hopkins University, Email: `mianjy@jhu.edu`

[§]Department of Applied Mathematics and Statistics, Johns Hopkins University, Email: `amukhe14@jhu.edu` to be computationally hard to learn in the worst case [2] and had mixed empirical success in early 1990s. Consequently, DNNs fell out of favor by late 90s.

Recently, there has been a resurgence of DNNs with the advent of deep learning [23]. Deep learning, loosely speaking, refers to a suite of computational techniques that have been developed recently for training DNNs. It started with the work of [15], which gave empirical evidence that if DNNs are initialized properly (for instance, using unsupervised pre-training), then we can find good solutions in a reasonable amount of runtime. This work was soon followed by a series of early successes of deep learning at significantly improving the state-of-the-art in speech recognition [14]. Since then, deep learning has received immense attention from the machine learning community with several state-of-the-art AI systems in speech recognition, image classification, and natural language processing based on deep neural nets [14, 7, 21, 22, 38]. While there is less of evidence now that pre-training actually helps, several other solutions have since been put forth to address the issue of efficiently training DNNs. These include heuristics such as dropouts [37], but also considering alternate deep architectures such as convolutional neural networks [32], deep belief networks [15], and deep Boltzmann machines [30]. In addition, deep architectures based on new non-saturating activation functions have been suggested to be more effectively trainable – the most successful and widely popular of these is the rectified linear unit (ReLU) activation, i.e., $\sigma(x) = \max\{0, x\}$, which is the focus of study in this paper.

In this paper, we formally study deep neural networks with rectified linear units; we refer to these deep architectures as ReLU DNNs. Our work is inspired by these recent attempts to understand the reason behind the successes of deep learning, both in terms of the structure of the functions represented by DNNs, [39, 40, 18, 34], as well as efforts which have tried to understand the non-convex nature of the training problem of DNNs better [19, 12]. Our investigation of the function space represented by ReLU DNNs also takes inspiration from the classical theory of circuit complexity; we refer the reader to [3, 35, 17, 31, 20, 1, 43, 5] for various surveys of this deep and fascinating field. In particular, our gap results are inspired by results like the ones by Hastad [13], Razborov [27] and Smolensky [36] which show a strict separation of complexity classes. We make progress towards similar statements with deep neural nets with ReLU activation.

2 Our Results and Techniques

2.1 Notation

We define a function $\sigma(x) = (\max\{0, x_1\}, \max\{0, x_2\}, \ldots, \max\{0, x_n\})$ for $x \in \mathbb{R}^n$.

Definition 1. [ReLU DNNs, depth, width, size] For any two natural numbers $n_1, n_2 \in \mathbb{N}$, which are called the *input* and *output dimensions* respectively, a $\mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ *ReLU DNN* is given by specifying a natural number $k \in \mathbb{N}$, a sequence of k natural numbers $w_1, w_2, \ldots, w_k$, a set of k affine transformations $T_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{w_1}$, $T_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i}$ for $i = 2, \ldots, k$ and a linear transformation $T_{k+1} : \mathbb{R}^{w_k} \to \mathbb{R}^{n_2}$. Such a ReLU DNN is called a $(k+1)$-layer ReLU DNN, and is said to have k hidden layers. The *function* $f : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ *computed or represented by this ReLU DNN* is

$$f = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ T_2 \circ \sigma \circ T_1, \tag{1}$$

where $\circ$ denotes function composition. The *depth* of a ReLU DNN is defined as $k+1$. The *width* of the i^{th} hidden layer is w_i, and the *width* of a ReLU DNN is $\max\{w_1, \ldots, w_k\}$. The *size* of the ReLU DNN is $w_1 + w_2 + \ldots + w_k$.

Definition 2 (Piecewise linear functions)**.** We say a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *continuous piecewise linear (PWL)* if there exists a *finite* set of polyhedra whose union is $\mathbb{R}^n$, and f is affine linear over each polyhedron (note that the definition automatically implies continuity of the function because the affine regions are closed and cover $\mathbb{R}^n$, and affine functions are continuous). The *number of pieces of f* is the number of maximal connected subsets of $\mathbb{R}^n$ over which f is affine linear (which is finite).

Many of our important statements will be phrased in terms of the following simplex.

Definition 3. Let $M > 0$ be any positive real number and $p \geq 1$ be any natural number. Define the set

$$\Delta_M^p = \{\mathbf{x} \in \mathbb{R}^p : 0 < \mathbf{x}_1 < \mathbf{x}_2 < \ldots < \mathbf{x}_p < M\}.$$

2.2 Statement of Results

2.2.1 Exact characterization of function class represented by ReLU DNNs

It is clear from definition that any function from $\mathbb{R}^n \to \mathbb{R}$ represented by a ReLU DNN is a continuous piecewise linear (PWL) function. The converse can be established using the following characterization of continuous piecewise linear functions by Wang and Sun [41]. We are extremely grateful to Dr. Chai Wah Wu and Dr. Robert Hildebrand at IBM for pointing us to this result.

Theorem 2.1. [Theorem 1 in [41]] For every piecewise linear function $f : \mathbb{R}^n \to \mathbb{R}$, there exists a finite set of affine linear functions $\ell_1, \ldots, \ell_k$ and subsets $S_1, \ldots, S_p \subseteq \{1, \ldots, k\}$ (not necessarily disjoint) where each S_i is of cardinality at most $n+1$, such that

$$f = \sum_{j=1}^{p} s_j \left(\max_{i \in S_j} \ell_i \right), \tag{2}$$

where $s_j \in \{-1, +1\}$ for all $j = 1, \ldots, p$.

Since a function of the form $\max_{i \in S_j} \ell_i$ is a piecewise linear convex function with at most $n+1$ pieces (because $|S_j| \leq n+1$), the above theorem says that any continuous piecewise linear function (not necessarily convex) can be obtained as a linear combination of piecewise linear convex functions each of which has at most $n+1$ affine pieces.

By Lemmas A.2 and A.3 in the Appendix (see supplementary material), we obtain the result that every $\mathbb{R}^n \to \mathbb{R}$ piecewise linear function can be expressed by a ReLU DNN with at most $\lceil \log_2(n+1) \rceil$ hidden layers, which we state formally in the following corollary.

Corollary 2.2. Every $\mathbb{R}^n \to \mathbb{R}$ ReLU DNN represents a piecewise linear function, and every piecewise linear function $\mathbb{R}^n \to \mathbb{R}$ can be represented by a ReLU DNN with at most $\lceil \log_2(n+1) \rceil + 1$ depth.

While Corollary 2.2 gives an upper bound on the depth of the networks needed to represent all continuous piecewise linear functions on $\mathbb{R}^n$, it does not give any tight bounds on the *size* of the networks that are needed to represent a given piecewise linear function. For $n = 1$, we give tight bounds on size as follows:

Theorem 2.3. Given any piecewise linear function $\mathbb{R} \to \mathbb{R}$ with p pieces there exists a 2-layer DNN with at most p nodes that can represent f. Moreover, any 2-layer DNN that represents f has size at least $p - 1$.

$L^q(\mathbb{R}^n)$ is the space of Lebesgue integrable functions f such that $\int |f|^q d\mu < \infty$, where μ is the Lebesgue measure on $\mathbb{R}^n$ (see Royden [29]). It is well known that the the family of compactly supported continuous functions are dense in $L^q(\mathbb{R}^n)$ and the piecewise linear functions are dense in the family of compactly supported continuous functions [29]). We combine these into the following statement.

Theorem 2.4. Every function in $L^q(\mathbb{R}^n)$, $(1 \leq q \leq \infty)$ can be arbitrarily well-approximated in the L^q norm (which for a function f is given by $||f||_q = (\int |f|^q)^{1/q})$ by a ReLU DNN function with at most $\lceil \log_2(n+1) \rceil$ hidden layers. Moreover, for $n = 1$, any such L^q function can be arbitrarily well-approximated by a 2-layer DNN, with tight bounds on the size of such a DNN in terms of the approximation.

A weaker version of Corollary 2.2 was also observed in [11, Proposition 4.1], where no bound on the depth was given. The authors of [11] also used a previous result of Wang [41] for obtaining their result. They also use this to obtain a universal approximation theorem similar to Theorem 2.4.

2.2.2 Training 2-layer $\mathbb{R} \to \mathbb{R}$ ReLU DNNs to global optimality

The *training problem* is the following: Given D data points $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, $i = 1, \ldots, D$, and a fixed topology for the ReLU DNNs, find the optimal set of weights/parameters for the DNN such that the corresponding function f represented by the DNN will minimize $\sum_{i=1}^{D} \ell(f(x_i), y_i)$, where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a *loss function* (common loss functions are the squared loss, $\ell(y, y') = (y - y')^2$, and the hinge loss function given by $\ell(y, y') = \max\{0, 1 - yy'\}$).

Theorem 2.5. There exists an algorithm to *optimally* train a 2-layer DNN of width w on D data points $(x_1, y_1), \ldots, (x_D, y_D) \in \mathbb{R}^n \times \mathbb{R}$, in time $O(2^w (D)^{nw} \text{poly}(D))$, i.e., the algorithm solves the following optimization problem to global optimality

$$\min_{T_1, T_2} \left\{ \sum_{i=1}^{D} \ell\big(\, T_2(\sigma(T_1(x_i))), y_i \,\big) : T_1 : \mathbb{R}^n \to \mathbb{R}^w \text{ affine }, T_2 : \mathbb{R}^w \to \mathbb{R} \text{ linear} \right\} \tag{3}$$

for any convex loss function $\ell : \mathbb{R} \to \mathbb{R}$. Note that the running time $O(2^w (D)^{nw} \text{poly}(D, n, w))$ is polynomial in the data size D for fixed n, w.

Key Insight: When the training problem is viewed as an optimization problem in the space of weights/parameters of the ReLU DNN, it is a nonconvex, quadratic problem similar to a matrix factorization problem. However, one can instead search over the space of functions representable by 2-layer DNNs by writing them in the form similar to (2). This breaks the problem into two parts: a combinatorial search and then a convex problem that is essentially linear regression with linear inequality constraints. This enables us to guarantee global optimality.

2.2.3 Circuit lower bounds for $\mathbb{R} \to \mathbb{R}$ ReLU DNNs

We extend a result of Telgarsky [39, 40] about the depth-size trade-off for ReLU DNNs.

Theorem 2.6. For every pair of natural numbers $k \geq 1$, $w \geq 2$, there exists a family of hard functions representable by a $\mathbb{R} \to \mathbb{R}$ $(k+1)$-layer ReLU DNN of width w such that if it is also representable by a $(k'+1)$-layer ReLU DNN for any $k' \leq k$, then this $(k'+1)$-layer ReLU DNN has size at least $\frac{1}{2}k'w^{\frac{k}{k'}} - 1$.

In fact our family of hard functions described above has a very intricate structure as stated below.

Theorem 2.7. For every $k \geq 1$, $w \geq 2$, every member of the family of hard functions in Theorem 2.6 has w^k pieces and this family can be parametrized by

$$\bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}}, \tag{4}$$

i.e., for every point in the set above, there exists a distinct function with the stated properties.

The following is an immediate corollary of Theorem 2.6 by choosing the parameters carefully.

Corollary 2.8. For every $k \in \mathbb{N}$ and $\epsilon > 0$, there is a family of functions defined on the real line such that every function f from this family can be represented by a $(k^{1+\epsilon}) + 1$-layer DNN with size $k^{2+\epsilon}$ and if f is represented by a $k+1$-layer DNN, then this DNN must have size at least $\frac{1}{2}k \cdot k^{k^\epsilon} - 1$. Moreover, this family can be parametrized as, $\cup_{M>0}\Delta_M^{k^{2+\epsilon}-1}$.

A particularly illuminative special case is obtained by setting $\epsilon = 1$ in Corollary 2.8:

Corollary 2.9. For every natural number $k \in \mathbb{N}$, there is a family of functions parameterized by the set $\cup_{M>0}\Delta_M^{k^3-1}$ such that any f from this family can be represented by a k^2+1-layer DNN with k^3 nodes, and every $k+1$-layer DNN that represents f needs at least $\frac{1}{2}k^{k+1} - 1$ nodes.

We can also get hardness of approximation versions of Theorem 2.6 and Corollaries 2.8 and 2.9, with the same gaps (upto constant terms), using the following theorem.

Theorem 2.10. For every $k \geq 1$, $w \geq 2$, there exists a function $f_{k,w}$ that can be represented by a $(k+1)$-layer ReLU DNN with w nodes in each layer, such that for all $\delta > 0$ and $k' \leq k$ the following holds:

$$\inf_{g \in \mathcal{G}_{k',\delta}} \int_{x=0}^{1} |f_{k,w}(x) - g(x)| dx > \delta,$$

where $\mathcal{G}_{k',\delta}$ is the family of functions representable by ReLU DNNs with depth at most $k'+1$, and size at most $\frac{w^{k/k'}(1-4\delta)^{1/k'}}{2^{1+1/k'}}k'$.

2.2.4 A continuum of hard functions for $\mathbb{R}^n \to \mathbb{R}$ for $n \geq 2$

One measure of complexity of a family of $\mathbb{R}^n \to \mathbb{R}$ "hard" functions represented by ReLU DNNs is the asymptotics of the number of pieces as a function of dimension n, depth $k+1$ and size s of the ReLU DNNs. More formally, suppose one has a family $\mathscr{F}$ of functions such that for every $n, k, s \in \mathbb{N}$ the family contains at least one $\mathbb{R}^n \to \mathbb{R}$ function representable by a ReLU DNN with depth at most $k+1$ and size at most s. Define the measure $\text{comp}_{\mathscr{F}}(n, k, s)$ as the maximum number of pieces of a $\mathbb{R}^n \to \mathbb{R}$ function from $\mathscr{F}$ that can be represented by a ReLU DNN with depth at most $k+1$ and size at most s. This measure has been studied in previous work [25, 26]. The best known families $\mathscr{F}$ are the ones from [25] and [39, 40] which achieve $\text{comp}_{\mathscr{F}}(n, k, s) = O(\frac{(s/k)^{kn}}{n^{n(k-1)}})$ and $\text{comp}_{\mathscr{F}}(n, k, s) = O(\frac{s^k}{k^k})$, respectively. We give the first construction that, for any *fixed* k, s, achieves an exponential dependence on n. In particular, we construct a class of functions for which $\text{comp}_{\mathscr{F}}(n, k, s) = \Omega(s^n)$. Moreover, for fixed n, k, s, these functions are smoothly parameterized. We make the formal statements below.

Theorem 2.11. For every tuple of natural numbers $n, k, m \geq 1$ and $w \geq 2$, there exists a family of $\mathbb{R}^n \to \mathbb{R}$ functions, which we call $\text{ZONOTOPE}^n_{k,w,m}$ with the following properties:

(i) Every $f \in \text{ZONOTOPE}^n_{k,w,m}$ is representable by a ReLU DNN of depth $k+2$ and size $2m+wk$, and has $(m-1)^{n-1}w^k$ pieces.

(ii) Consider any $f \in \text{ZONOTOPE}^n_{k,w,m}$. If f is represented by a $(k'+1)$-layer DNN for any $k' \leq k$, then this $(k'+1)$-layer DNN has size at least $\max\left\{\frac{1}{2}(k' w^{\frac{k}{k'n}}) \cdot (m-1)^{(1-\frac{1}{n})\frac{1}{k'}} - 1 \;,\; \frac{w^{\frac{k}{k'}}}{n^{1/k'}} k'\right\}$.

(iii) The family $\text{ZONOTOPE}^n_{k,w,m}$ is in one-to-one correspondence with

$$S(n,m) \times \bigcup_{M>0} \underbrace{(\Delta^{w-1}_M \times \Delta^{w-1}_M \times \ldots \times \Delta^{w-1}_M)}_{k \text{ times}},$$

where $S(n,m)$ is the so-called "extremal zonotope set", which is a subset of $\mathbb{R}^{nm}$, whose complement has zero Lebesgue measure in $\mathbb{R}^{nm}$.

To obtain $\text{comp}_{\mathscr{F}}(n, k, s) = \Omega(s^n)$ from the above statement, we find $w \in \mathbb{N}$ such that $s \geq w^{k-1} + w(k-1)$ and set $m = w^{k-1}$. Then the functions in $\text{ZONOTOPE}^n_{k,w,m}$ have $\sim s^n$ pieces.

2.3 Roadmap

The proofs of the above theorems are arranged as follows. Theorems 2.3 and 2.4 are proved in Section 3. Theorem 2.5 is in Section 4. Theorems 2.6 and 2.7 are proved in Section 5. Theorem 2.10 is proved in Section 5.1. Theorem 2.11 is proved in Section 6. Some auxiliary results and structural lemmas appear in Appendix A.

2.4 Comparing our training results with previous work

As mentioned before, a weaker version of Corollary 2.2 was observed in [11, Proposition 4.1] (with no bound on the depth), along with a universal approximation theorem [11, Theorem 4.3] similar to Theorem 2.4. However, we are not aware of tight bounds like Theorem 2.3 in previous work on ReLU DNNs.

To the best of our knowledge, the training algorithm from Section 2.2.2 is the first in the literature of training neural nets that has a guarantee of reaching the global optimum. Moreover, it does so in time polynomial in the number of data points, if the size of the network being trained and the number of inputs are thought of as a fixed constants. To the best of our knowledge there is no hardness result known which rules out empirical risk minimization of deep nets in time polynomial in circuit size or data size. Thus our training result is a step towards resolving this gap in the complexity literature. Moreover, for $n = 1$, the algorithm is based on the characterization from Theorem 2.3, which underscores the usefulness of that result. A related algorithm for *improperly* learning ReLU networks with a single hidden layer has been recently obtained by Goel et al [10], with running time $n^{O(1)}2^{O(\sqrt{w}/\epsilon)}$, where ϵ is the error bound on the expected loss (n, w are the number of inputs and size of the network, respectively, just like in this paper). In contrast, our algorithm is a *proper* learning algorithm.

2.5 Comparing our hardness results with previous work

The results on depth-size trade-off from Section 2.2.3 have recent predecessors in Telgarsky's elegant theorems from [39, 40]. Our results extend and improve Telgarsky's achievements in three ways:

(i) If we use our Theorem 2.10 to the pair of neural nets considered by Telgarsky in Theorem 1.1 in [40] which are at depths k^3 (of size also scaling as k^3) and k then for this purpose of approximation in the ℓ_1-norm we would get a size lower bound for the shallower net which scales as $\Omega(2^{k^2})$ which is exponentially (in depth) larger than the lower bound of $\Omega(2^k)$ that Telgarsky can get for this scenario.

(ii) Telgarsky's family of hard functions is parameterized by a single natural number k. In contrast, we show that for every *pair* of natural numbers w and k, and a point from the set in equation 4, there exists a "hard" function which to be represented by a depth k' network would need a size of at least $w^{\frac{k}{k'}}k'$. With the extra flexibility of choosing the parameter w, for the purpose of showing gaps in representation ability of deep nets we can shows size lower bounds which are *super*-exponential in depth as explained in Corollaries 2.8 and 2.9.

(iii) A characteristic feature of the "hard" functions in Boolean circuit complexity is that they are usually a countable family of functions and not a "smooth" family of hard functions. In fact, in the last section of [39], Telgarsky states this as a "weakness" of the state-of-the-art results on "hard" functions for both Boolean circuit complexity and neural nets research. In contrast, we provide a smoothly parameterized family of "hard" functions in Section 2.2.3 (parametrized by the set in equation 4). Such a continuum of hard functions wasn't demonstrated before this work.

We point out that Telgarsky's results in [40] apply to deep neural nets with a host of different activation functions, whereas, our results are specifically for neural nets with rectified linear units. In this sense, Telgarsky's results from [40] are more general than our results in this paper, but with weaker gap guarantees. Eldan-Shamir [34, 9] show that there exists an $\mathbb{R}^n \to \mathbb{R}$ function that can be represented by a 3-layer DNN, that takes exponential in n number of nodes to be approximated to within some constant by a 2-layer DNN. While their results are not immediately comparable with Telgarsky's or our results, it is an interesting open question to extend their results to a constant depth hierarchy statement analogous to the recent breakthrough of Rossman et al [28].

3 Every $\mathbb{R} \to \mathbb{R}$ CPWL function is representable by 2-layer ReLU DNNs

Proof of Theorem 2.3. Any continuous piecewise linear function $\mathbb{R} \to \mathbb{R}$ which has m pieces can be specified by three pieces of information, (1) s_L the slope of the left most piece, (2) the coordinates of the non-differentiable points specified by a $(m-1)-$tuple $\{(a_i, b_i)\}_{i=1}^{m-1}$ (indexed from left to right) and (3) s_R the slope of the rightmost piece. A tuple $(s_L, s_R, (a_1, b_1), \ldots, (a_{m-1}, b_{m-1})$ uniquely specifies a m piecewise linear function from $\mathbb{R} \to \mathbb{R}$ and vice versa. Given such a tuple, we construct a 2-layer DNN which computes the same piecewise linear function.

One notes that for any $a, r \in \mathbb{R}$, the function

$$f(x) = \begin{cases} 0 & x \leq a \\ r(x-a) & x > a \end{cases} \tag{5}$$

is equal to $\operatorname{sgn}(r)\max\{|r|(x-a), 0\}$, which can be implemented by a 2-layer ReLU DNN with size 1. Similarly, any function of the form,

$$g(x) = \begin{cases} t(x-a) & x \leq a \\ 0 & x > a \end{cases} \tag{6}$$

is equal to $-\operatorname{sgn}(t)\max\{-|t|(x-a), 0\}$, which can be implemented by a 2-layer ReLU DNN with size 1. The parameters r, t will be called the *slopes* of the function, and a will be called the *breakpoint* of the function.If we can write the given piecewise linear function as a sum of m functions of the form (5) and (6), then by Lemma A.2 we would be done. It turns out that such a decomposition of any p piece PWL function $h : \mathbb{R} \to \mathbb{R}$ as a sum of p flaps can always be arranged where the breakpoints of the p flaps all are all contained in the $p-1$ breakpoints of h. First, observe that adding a constant to a function does not change the complexity of the ReLU DNN expressing it, since this corresponds to a bias on the output node. Thus, we will assume that the value of h at the last break point a_{m-1} is $b_{m-1} = 0$. We now use a single function f of the form (5) with slope r and breakpoint $a = a_{m-1}$, and $m-1$ functions $g_1, \ldots, g_{m-1}$ of the form (6) with slopes $t_1, \ldots, t_{m-1}$ and breakpoints $a_1, \ldots, a_{m-1}$, respectively. Thus, we wish to express $h = f + g_1 + \ldots + g_{m-1}$. Such a decomposition of h would be valid if we can find values for $r, t_1, \ldots, t_{m-1}$ such that (1) the slope of the above sum is $= s_L$ for $x < a_1$, (2) the slope of the above sum is $= s_R$ for $x > a_{m-1}$, and (3) for

each $i \in \{1, 2, 3, .., m-1\}$ we have $b_i = f(a_i) + g_1(a_i) + \ldots + g_{m-1}(a_i)$.
The above corresponds to asking for the existence of a solution to the following set of simultaneous linear equations in $r, t_1, \ldots, t_{m-1}$:

$$s_R = r, \;\; s_L = t_1 + t_2 + \ldots + t_{m-1}, \;\; b_i = \sum_{j=i+1}^{m-1} t_j(a_{j-1} - a_j) \text{ for all } i = 1, \ldots, m-2$$

It is easy to verify that the above set of simultaneous linear equations has a unique solution. Indeed, r must equal s_R, and then one can solve for $t_1, \ldots, t_{m-1}$ starting from the last equation $b_{m-2} = t_{m-1}(a_{m-2} - a_{m-1})$ and then back substitute to compute $t_{m-2}, t_{m-3}, \ldots, t_1$. The lower bound of $p-1$ on the size for any 2-layer ReLU DNN that expresses a p piece function follows from Lemma A.6. □

One can do better in terms of size when the rightmost piece of the given function is flat, i.e., $s_R = 0$. In this case $r = 0$, which means that $f = 0$; thus, the decomposition of h above is of size $p-1$. A similar construction can be done when $s_L = 0$. This gives the following statement which will be useful for constructing our forthcoming hard functions.

Corollary 3.1. If the rightmost or leftmost piece of a $\mathbb{R} \to \mathbb{R}$ piecewise linear function has 0 slope, then we can compute such a p piece function using a 2-layer DNN with size $p-1$.

Proof of theorem 2.4. Since any piecewise linear function $\mathbb{R}^n \to \mathbb{R}$ is representable by a ReLU DNN by Corollary 2.2, the proof simply follows from the fact that the family of continuous piecewise linear functions is dense in any $L^p(\mathbb{R}^n)$ space, for $1 \leq p \leq \infty$. □

4 Exact empirical risk minimization for the 2−layer $\mathbb{R}^n \to \mathbb{R}$ ReLU DNNs

Proof of Theorem 2.5. Let $\ell : \mathbb{R} \to \mathbb{R}$ be any convex loss function, and let $(x_1, y_1), \ldots, (x_D, y_D) \in \mathbb{R}^n \times \mathbb{R}$ be the given D data points. As stated in (3), the problem requires us to find an affine transformation $T_1 : \mathbb{R}^n \to \mathbb{R}^w$ and a linear transformation $T_2 : \mathbb{R}^w \to \mathbb{R}$, so as to minimize the empirical loss as stated in (3). Note that T_1 is given by a matrix $A \in \mathbb{R}^{w \times n}$ and a vector $b \in \mathbb{R}^w$ so that $T(x) = Ax + b$ for all $x \in \mathbb{R}^n$. Similarly, T_2 can be represented by a vector $a' \in \mathbb{R}^w$ such that $T_2(y) = a' \cdot y$ for all $y \in \mathbb{R}^w$. If we denote the i-th row of the matrix A by a^i, and write b_i, a'_i to denote the i-th coordinates of the vectors b, a' respectively, we can write the function represented by this network as

$$f(x) = \sum_{i=1}^{w} a'_i \max\{0, a^i \cdot x + b_i\} = \sum_{i=1}^{w} \text{sgn}(a'_i) \max\{0, (|a'_i|a^i) \cdot x + |a'_i|b_i\}.$$

In other words, the family of functions over which we are searching is of the form

$$f(x) = \sum_{i=1}^{w} s_i \max\{0, \tilde{a}^i \cdot x + \tilde{b}_i\} \tag{7}$$

where $\tilde{a}^i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ and $\mathrm{s}_i \in \{-1,+1\}$ for all $i = 1,\dots,w$. We now make the following observation. For a given data point (x_j, y_j) if $\tilde{a}^i \cdot x_j + \tilde{b}_i \le 0$, then the i-th term of (7) does not contribute to the loss function for this data point (x_j, y_j). Thus, for every data point (x_j, y_j), there exists a set $S_j \subseteq \{1,\dots,w\}$ such that $f(x_j) = \sum_{i \in S_j} s_i(\tilde{a}^i \cdot x_j + \tilde{b}_i)$. In particular, if we are given the set S_j for (x_j, y_j), then the expression on the right hand side of (7) reduces to a linear function of $\tilde{a}^i, \tilde{b}_i$. For any fixed $i \in \{1,\dots,w\}$, these sets S_j induce a partition of the data set into two parts. In particular, we define $P^i_+ := \{j : i \in S_j\}$ and $P^i_- := \{1,\dots,D\} \setminus P^i_+$. Observe now that this partition is also induced by the hyperplane given by $\tilde{a}^i, \tilde{b}_i$: $P^i_+ = \{j : \tilde{a}^i \cdot x_j + \tilde{b}_i > 0\}$ and $P^i_+ = \{j : \tilde{a}^i \cdot x_j + \tilde{b}_i \le 0\}$. Our strategy will be to *guess* the partitions P^i_+, P^i_- for each $i = 1,\dots,w$, and then do linear regression with the constraint that regression's decision variables $\tilde{a}^i, \tilde{b}_i$ induce the guessed partition.

More formally, the algorithm does the following. For each $i = 1,\dots,w$, the algorithm guesses a partition of the data set (x_j, y_j), $j = 1,\dots,D$ by a hyperplane. Let us label the partitions as follows (P^i_+, P^i_-), $i = 1,\dots,w$. So, for each $i = 1,\dots,w$, $P^i_+ \cup P^i_- = \{1,\dots,D\}$, P^i_+ and P^i_- are disjoint, and there exists a vector $c \in \mathbb{R}^n$ and a real number δ such that $P^i_- = \{j : c \cdot x_j + \delta \le 0\}$ and $P^i_+ = \{j : c \cdot x_j + \delta > 0\}$. Further, for each $i = 1,\dots,w$ the algorithm selects a vector s in $\{+1,-1\}^w$.

For a fixed selection of partitions (P^i_+, P^i_-), $i = 1,\dots,w$ and a vector s in $\{+1,-1\}^w$, the algorithm solves the following convex optimization problem with decision variables $\tilde{a}^i \in \mathbb{R}^n$, $\tilde{b}_i \in \mathbb{R}$ for $i = 1,\dots,w$ (thus, we have a total of $(n+1)\cdot w$ decision variables). The feasible region of the optimization is given by the constraints

$$\begin{aligned} \tilde{a}^i \cdot x_j + \tilde{b}_i \le 0 \quad \forall j \in P^i_- \\ \tilde{a}^i \cdot x_j + \tilde{b}_i \ge 0 \quad \forall j \in P^i_+ \end{aligned} \tag{8}$$

which are imposed for all $i = 1,\dots,w$. Thus, we have a total of $D \cdot w$ constraints. Subject to these constraints we minimize the objective $\sum_{j=1}^{D} \sum_{i: j \in P^i_+} \ell(s_i(\tilde{a}^i \cdot x_j + \tilde{b}_i), y_j)$. Assuming the loss function ℓ is a convex function in the first argument, the above objective is a convex function. Thus, we have to minize a convex objective subject to the linear inequality constraints from (8).

We finally have to count how many possible partitions (P^i_+, P^i_-) and vectors s the algorithm has to search through. It is well-known [24] that the total number of possible hyperplane partitions of a set of size D in $\mathbb{R}^n$ is at most $2\binom{D}{n} \le D^n$ whenever $n \ge 2$. Thus with a guess for each $i = 1,\dots,w$, we have a total of at most D^{nw} partitions. There are 2^w vectors s in $\{-1,+1\}^w$. This gives us a total of $2^w D^{nw}$ guesses for the partitions (P^i_+, P^i_-) and vectors s. For each such guess, we have a convex optimization problem with $(n+1)\cdot w$ decision variables and $D \cdot w$ constraints, which can be solved in time $\mathrm{poly}(D, n, w)$. Putting everything together, we have the running time claimed in the statement.

The above argument holds only for $n \ge 2$, since we used the inequality $2\binom{D}{n} \le D^n$ which only holds for $n \ge 2$. For $n = 1$, a similar algorithm can be designed, but one which uses the characterization achieved in Theorem 2.3. We discuss in the next section. $\square$

4.1 Training the 2-layer network with a single input

Let $\ell : \mathbb{R} \to \mathbb{R}$ be any convex loss function, and let $(x_1, y_1), \ldots, (x_D, y_D) \in \mathbb{R}^2$ be the given D data points. Using Theorem 2.3, to solve problem (3) it suffices to find a $\mathbb{R} \to \mathbb{R}$ piecewise linear function f with w pieces that minimizes the total loss. In other words, the optimization problem (3) is equivalent to the problem

$$\min \left\{ \sum_{i=1}^{D} \ell(f(x_i), y_i) : f \text{ is piecewise linear with } w \text{ pieces} \right\}. \tag{9}$$

We now use the observation that fitting piecewise linear functions to minimize loss is just a step away from linear regression, which is a special case where the function is contrained to have exactly one affine linear piece. Our algorithm will first guess the optimal partition of the data points such that all points in the same class of the partition correspond to the same affine piece of f, and then do linear regression in each class of the partition. Altenatively, one can think of this as guessing the interval (x_i, x_{i+1}) of data points where the $w-1$ breakpoints of the piecewise linear function will lie, and then doing linear regression between the breakpoints.

More formally, we parametrize piecewise linear functions with w pieces by the w slope-intercept values $(a_1, b_1), \ldots, (a_2, b_2), \ldots, (a_w, b_w)$ of the w different pieces. This means that between breakpoints j and $j+1$, $1 \leq j \leq w-2$, the function is given by $f(x) = a_{j+1}x + b_{j+1}$, and the first and last pieces are $a_1 x + b_1$ and $a_w x + b_w$, respectively.

Define $\mathscr{I}$ to be the set of all $(w-1)$-tuples $(i_1, \ldots, i_{w-1})$ of natural numbers such that $1 \leq i_1 \leq \ldots \leq i_{w-1} \leq D$. Given a fixed tuple $I = (i_1, \ldots, i_{w-1}) \in \mathscr{I}$, we wish to search through all piecewise linear functions whose breakpoints, in order, appear in the intervals $(x_{i_1}, x_{i_1+1}), (x_{i_2}, x_{i_2+1}), \ldots, (x_{i_{w-1}}, x_{i_{w-1}+1})$. Define also $\mathscr{S} = \{-1, 1\}^{w-1}$. Any $S \in \mathscr{S}$ will have the following interpretation: if $S_j = 1$ then $a_j \leq a_{j+1}$, and if $S_j = -1$ then $a_j \geq a_{j+1}$. Now for every $I \in \mathscr{I}$ and $S \in \mathscr{S}$, requiring a piecewise linear function that respects the conditions imposed by I and S is easily seen to be equivalent to imposing the following linear inequalities on the parameters $(a_1, b_1), \ldots, (a_2, b_2), \ldots, (a_w, b_w)$:

$$\begin{aligned} S_j(b_{j+1} - b_j - (a_j - a_{j+1})x_{i_j}) &\geq 0 \\ S_j(b_{j+1} - b_j - (a_j - a_{j+1})x_{i_j+1}) &\leq 0 \\ S_j(a_{j+1} - a_j) &\geq 0 \end{aligned} \tag{10}$$

Let the set of piecewise linear functions whose breakpoints satisfy the above be denoted by $\mathrm{PWL}^1_{I,S}$ for $I \in \mathscr{I}, S \in \mathscr{S}$.

Given a particular $I \in \mathscr{I}$, we define

$$\begin{aligned} &D_1 := \{x_i : i \leq i_1\}, \\ &D_j := \{x_i : i_{j-1} < i \leq i_1\} \quad j = 2, \ldots, w-1, \ . \\ &D_w := \{x_i : i > i_{w-1}\} \end{aligned}$$

Observe that

$$\min\{\sum_{i=1}^{D} \ell(f(x_i) - y_i) : f \in \mathrm{PWL}^1_{I,S}\} = \min\{\sum_{j=1}^{w} \left(\sum_{i \in D_j} \ell(a_j \cdot x_i + b_j - y_i) \right) : (a_j, b_j) \text{ satisfy } (10)\} \tag{11}$$

The right hand side of the above equation is the problem of minimizing a convex objective subject to linear constraints. Now, to solve (9), we need to simply solve the problem (11) for all $I \in \mathscr{I}, S \in \mathscr{S}$ and pick the minimum. Since $|\mathscr{I}| = \binom{D}{w} = O(D^w)$ and $|\mathscr{S}| = 2^{w-1}$ we need to solve $O(2^w \cdot D^w)$ convex optimization problems, each taking time $O(\text{poly}(D))$. Therefore, the total running time is $O((2D)^w \text{poly}(D))$.

5 Constructing a continuum of hard functions for $\mathbb{R} \to \mathbb{R}$ ReLU DNNs at every depth and every width

Definition 4. Let $M > 0$ be a given real number, and let $p \in \mathbb{N}$. Let $\mathbf{a} \in \Delta_M^p$. We define a function $h_{\mathbf{a}} : \mathbb{R} \to \mathbb{R}$ which is piecewise linear over the segments $(-\infty, 0], [0, \mathbf{a}_1], [\mathbf{a}_1, \mathbf{a}_2], \ldots, [\mathbf{a}_p, M], [M, +\infty)$ defined as follows: $h_{\mathbf{a}}(x) = 0$ for all $x \leq 0$, $h_{\mathbf{a}}(\mathbf{a}_i) = M(i \mod 2)$, and $h_{\mathbf{a}}(1) = M - h_{\mathbf{a}}(\mathbf{a}_p)$ and for $x \geq 1$, $h_{\mathbf{a}}(x)$ is a linear continuation of the piece over the interval $[\mathbf{a}_p, M]$. Note that the function has $p + 2$ pieces, with the leftmost piece having slope 0.

Lemma 5.1. For any $M > 0$, $p \in \mathbb{N}$, $k \in \mathbb{N}$ and $\mathbf{a}^1, \ldots, \mathbf{a}^k \in \Delta_M^p$, if we compose the functions $h_{\mathbf{a}^1}, h_{\mathbf{a}^2}, \ldots, h_{\mathbf{a}^k}$ the resulting function is a piecewise linear function with at most $(p+1)^k + 2$ pieces, i.e.,

$$H_{\mathbf{a}^1,\ldots,\mathbf{a}^k} := h_{\mathbf{a}^k} \circ h_{\mathbf{a}^{k-1}} \circ \ldots \circ h_{\mathbf{a}^1}$$

is piecewise linear with at most $(p+1)^k + 2$ pieces, with $(p+1)^k$ of these pieces in the range $[0, M]$ (see Figure 1). Moreover, in each piece in the range $[0, M]$, the function is affine with minimum value 0 and maximum value M.

Proof. Simple induction on k. $\square$

Proof of Theorem 2.7. Given $k \geq 1$ and $w \geq 2$, choose any point

$$(\mathbf{a}^1, \ldots, \mathbf{a}^k) \in \bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}}.$$

By Definition 4, each $h_{\mathbf{a}^i}$, $i = 1, \ldots, k$ is a piecewise linear function with $w + 1$ pieces and the leftmost piece having slope 0. Thus, by Corollary 3.1, each $h_{\mathbf{a}^i}$, $i = 1, \ldots, k$ can be represented by a 2-layer ReLU DNN with size w. Using Lemma A.1, $H_{\mathbf{a}^1,\ldots,\mathbf{a}^k}$ can be represented by a $k + 1$ layer DNN with size wk; in fact, each hidden layer has exactly w nodes. $\square$

Proof of Theorem 2.6. Follows from Theorem 2.7 and Lemma A.6. $\square$

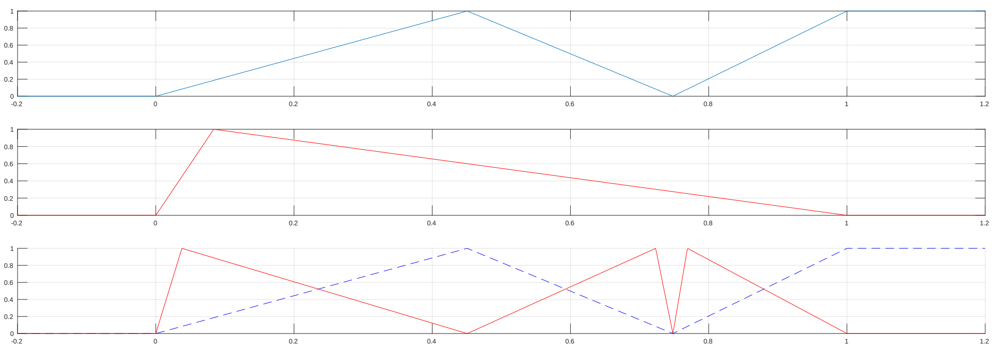


Figure 1: Top: $h_{\mathbf{a}^1}$ with $\mathbf{a}^1 \in \Delta_1^2$ with 3 pieces in the range $[0,1]$. Middle: $h_{\mathbf{a}^2}$ with $\mathbf{a}^2 \in \Delta_1^1$ with 2 pieces in the range $[0,1]$. Bottom: $H_{\mathbf{a}^1,\mathbf{a}^2} = h_{\mathbf{a}^2} \circ h_{\mathbf{a}^1}$ with $2 \cdot 3 = 6$ pieces in the range $[0,1]$. The dotted line in the bottom panel corresponds to the function in the top panel. It shows that for every piece of the dotted graph, there is a full copy of the graph in the middle panel.

5.1 L^1-norm gap analysis

Proof of Theorem 2.10. Given $k \geq 1$ and $w \geq 2$ define $q := w^k$ and $s_q := \underbrace{h_{\mathbf{a}} \circ h_{\mathbf{a}} \circ \ldots \circ h_{\mathbf{a}}}_{k \text{ times}}$ where $\mathbf{a} = (\frac{1}{w}, \frac{2}{w}, \ldots, \frac{w-1}{w}) \in \Delta_1^{q-1}$. Thus, s_q is representable by a ReLU DNN of width $w+1$ and depth $k+1$ by Lemma A.1. In what follows, we want to give a lower bound on the ℓ^1 distance of s_q from any continuous p-piecewise linear comparator $g_p : \mathbb{R} \to \mathbb{R}$. The function s_q contains $\lfloor \frac{q}{2} \rfloor$ triangles of width $\frac{2}{q}$ and unit height. A p-piecewise linear function has $p-1$ breakpoints in the interval $[0,1]$. So that in at least $\lfloor \frac{w^k}{2} \rfloor - (p-1)$ triangles, g_p has to be affine. In the following we demonstrate that inside any triangle of s_q, any affine function will incur an ℓ^1 error of at least $\frac{1}{2w^k}$.

$$\begin{aligned}
\int_{x=\frac{2i}{w^k}}^{\frac{2i+2}{w^k}} |s_q(x) - g_p(x)| dx &= \int_{x=0}^{\frac{2}{w^k}} \left| s_q(x) - (y_1 + (x-0) \cdot \frac{y_2 - y_1}{\frac{2}{w^k} - 0}) \right| dx \\
&= \int_{x=0}^{\frac{1}{w^k}} \left| xw^k - y_1 - \frac{w^k x}{2}(y_2 - y_1) \right| dx + \int_{x=\frac{1}{w^k}}^{\frac{2}{w^k}} \left| 2 - xw^k - y_1 - \frac{w^k x}{2}(y_2 - y_1) \right| dx \\
&= \frac{1}{w^k} \int_{z=0}^{1} \left| z - y_1 - \frac{z}{2}(y_2 - y_1) \right| dz + \frac{1}{w^k} \int_{z=1}^{2} \left| 2 - z - y_1 - \frac{z}{2}(y_2 - y_1) \right| dz \\
&= \frac{1}{w^k} \left(-3 + y_1 + \frac{2y_1^2}{2 + y_1 - y_2} + y_2 + \frac{2(-2+y_1)^2}{2 - y_1 + y_2} \right)
\end{aligned}$$

The above integral attains its minimum of $\frac{1}{2w^k}$ at $y_1 = y_2 = \frac{1}{2}$. Putting together,

$$\|s_{w^k} - g_p\|_1 \geq \left(\lfloor \frac{w^k}{2} \rfloor - (p-1) \right) \cdot \frac{1}{2w^k} \geq \frac{w^k - 1 - 2(p-1)}{4w^k} = \frac{1}{4} - \frac{2p-1}{4w^k}$$

Thus, for any $\delta > 0$,

$$p \leq \frac{w^k - 4w^k\delta + 1}{2} \implies 2p - 1 \leq (\frac{1}{4} - \delta)4w^k \implies \frac{1}{4} - \frac{2p-1}{4w^k} \geq \delta \implies \|s_{w^k} - g_p\|_1 \geq \delta.$$

The result now follows from Lemma A.6. □

6 Constructing a continuum of hard functions for $\mathbb{R}^n \to \mathbb{R}$ ReLU DNNs at every depth

Definition 5. Let $\mathbf{b}^1, \ldots, \mathbf{b}^m \in \mathbb{R}^n$. The zonotope formed by $\mathbf{b}^1, \ldots, \mathbf{b}^m \in \mathbb{R}^n$ is defined as

$$Z(\mathbf{b}^1, \ldots, \mathbf{b}^m) := \{\lambda_1 \mathbf{b}^1 + \ldots + \lambda_m \mathbf{b}^m : -1 \leq \lambda_i \leq 1, \ \ i = 1, \ldots, m\}.$$

The set of vertices of $Z(\mathbf{b}^1, \ldots, \mathbf{b}^m)$ will be denoted by $\operatorname{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))$. The *support function* $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)} : \mathbb{R}^n \to \mathbb{R}$ associated with the zonotope $Z(\mathbf{b}^1, \ldots, \mathbf{b}^m)$ is defined as

$$\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = \max_{\mathbf{x} \in Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)} \langle \mathbf{r}, \mathbf{x} \rangle.$$

The following results are well-known in the theory of zonotopes [42].

Theorem 6.1. The following are all true.

1. $|\operatorname{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))| \leq (m-1)^{n-1}$. The set of $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in \mathbb{R}^n \times \ldots \times \mathbb{R}^n$ such that this DOES NOT hold at equality is a 0 measure set.

2. $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = \max_{\mathbf{x} \in Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)} \langle \mathbf{r}, \mathbf{x} \rangle = \max_{\mathbf{x} \in \operatorname{vert}(Z(\mathbf{b}^1,\ldots,\mathbf{b}^m))} \langle \mathbf{r}, \mathbf{x} \rangle$, and $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}$ is therefore a piecewise linear function with $|\operatorname{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))|$ pieces.

3. $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = |\langle \mathbf{r}, \mathbf{b}^1 \rangle| + \ldots + |\langle \mathbf{r}, \mathbf{b}^m \rangle|$.

Definition 6. The set $S(n, m)$ will denote the set of $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in \mathbb{R}^n \times \ldots \times \mathbb{R}^n$ such that $|\operatorname{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))| = (m-1)^{n-1}$.

Lemma 6.2. Given any $\mathbf{b}^1, \ldots, \mathbf{b}^m \in \mathbb{R}^n$, there exists a 2-layer ReLU DNN with size $2m$ which represents the function $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r})$.

Proof. By Theorem 6.1 part 3., $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = |\langle \mathbf{r}, \mathbf{b}^1 \rangle| + \ldots + |\langle \mathbf{r}, \mathbf{b}^m \rangle|$. It suffices to observe

$$|\langle \mathbf{r}, \mathbf{b}^1 \rangle| + \ldots + |\langle \mathbf{r}, \mathbf{b}^m \rangle| = \max\{\langle \mathbf{r}, \mathbf{b}^1 \rangle, -\langle \mathbf{r}, \mathbf{b}^1 \rangle\} + \ldots + \max\{\langle \mathbf{r}, \mathbf{b}^m \rangle, -\langle \mathbf{r}, \mathbf{b}^m \rangle\}.$$

□

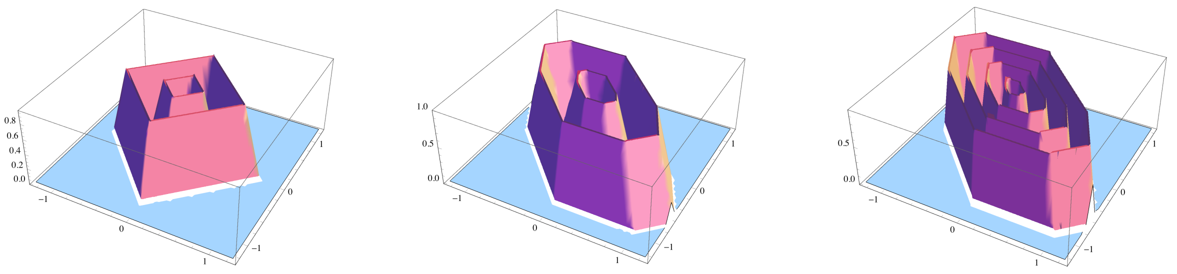
Proposition 6.3. Given any tuple $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in S(n, m)$ and any point

$$(\mathbf{a}^1, \ldots, \mathbf{a}^k) \in \bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}},$$

the function $\text{ZONOTOPE}^n_{k,w,m}[\mathbf{a}^1, \ldots, \mathbf{a}^k, \mathbf{b}^1, \ldots, \mathbf{b}^m] := H_{\mathbf{a}^1,\ldots,\mathbf{a}^k} \circ \gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}$ has $(m-1)^{n-1} w^k$ pieces and it can be represented by a $k+2$ layer ReLU DNN with size $2m + wk$.

Proof. The fact that $\text{ZONOTOPE}^n_{k,w,m}[\mathbf{a}^1, \ldots, \mathbf{a}^k, \mathbf{b}^1, \ldots, \mathbf{b}^m]$ can be represented by a $k+2$ layer ReLU DNN with size $2m + wk$ follows from Lemmas 6.2 and A.1. The number of pieces follows from the fact that $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}$ has $(m-1)^{n-1}$ distinct linear pieces by parts 1. and 2. of Theorem 6.1, and $H_{\mathbf{a}^1,\ldots,\mathbf{a}^k}$ has w^k pieces by Lemma 5.1. □

Proof of Theorem 2.11. Follows from Proposition 6.3. □



(a) $H_{\frac{1}{2},\frac{1}{2}} \circ N_{\ell_1}$ (b) $H_{\frac{1}{2},\frac{1}{2}} \circ \gamma_{Z(\mathbf{b}^1,\mathbf{b}^2,\mathbf{b}^3,\mathbf{b}^4)}$ (c) $H_{\frac{1}{2},\frac{1}{2},\frac{1}{2}} \circ \gamma_{Z(\mathbf{b}^1,\mathbf{b}^2,\mathbf{b}^3,\mathbf{b}^4)}$

Figure 2: We fix the $\mathbf{a}$ vectors for a two hidden layer $\mathbb{R} \to \mathbb{R}$ hard function as $\mathbf{a}^1 = \mathbf{a}^2 = (\frac{1}{2}) \in \Delta_1^1$ Left: A specific hard function induced by ℓ_1 norm: $\text{ZONOTOPE}^2_{2,2,2}[\mathbf{a}^1, \mathbf{a}^2, \mathbf{b}^1, \mathbf{b}^2]$ where $\mathbf{b}^1 = (0, 1)$ and $\mathbf{b}^2 = (1, 0)$. Note that in this case the function can be seen as a composition of $H_{\mathbf{a}^1,\mathbf{a}^2}$ with ℓ_1-norm $N_{\ell_1}(x) := \|x\|_1 = \gamma_{Z((0,1),(1,0))}$. Middle: A typical hard function $\text{ZONOTOPE}^2_{2,2,4}[\mathbf{a}^1, \mathbf{a}^2, \mathbf{c}^1, \mathbf{c}^2, \mathbf{c}^3, \mathbf{c}^4]$ with generators $\mathbf{c}^1 = (\frac{1}{4}, \frac{1}{2}), \mathbf{c}^2 = (-\frac{1}{2}, 0), \mathbf{c}^3 = (0, -\frac{1}{4})$ and $\mathbf{c}^4 = (-\frac{1}{4}, -\frac{1}{4})$. Note how increasing the number of zonotope generators makes the function more complex. Right: A *harder* function from $\text{ZONOTOPE}^2_{3,2,4}$ family with the same set of generators $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, c_4$ but one more hidden layer ($k = 3$). Note how increasing the depth make the function more complex. (For illustrative purposes we plot only the part of the function which lies above zero.)

7 Discussion

The running time of the algorithm that we give in this work to find the exact global minima of a two layer ReLU-DNN is exponential in the input dimension n and the number of hidden nodes w. It is unlikely, due to complexity theory beliefs such as $P \neq NP$, that the exponential dependence on n can be removed, if one requires the global optimality guarantee; see the book by Shalev-Schwartz and Ben-David [33] and [4, 8]. However, it is not clear to us whether the globally optimal algorithms have to necessarily be exponential in the number of hidden nodes. We are not aware of any complexity results which would rule out the possibility of an algorithm which trains to global optimality in time that is polynomial in the data size and the number of hidden nodes, assuming that the input dimension is a fixed constant. Resolving this dependence on network size would be another step towards clarifying the theoretical complexity of training ReLU DNNs and is a good open question for future research, in our opinion. Perhaps an even better breakthrough would be to get optimal training algorithms for DNNs with two or more hidden layers and this seems like a substantially harder nut to crack. It would also be a significant breakthrough to get gap results between consecutive constant depths or between logarithmic and constant depths.

Acknowledgements

Anirbit would like to thank Ramprasad Saptharishi, Piyush Srivastava and Rohit Gurjar for extensive discussions on Boolean and arithmetic circuit complexity. This paper has been immensely influenced by the perspectives gained during those extremely helpful discussions. Amitabh Basu gratefully acknowledges support from the NSF grant CMMI1452820. Raman Arora was supported in part by NSF BIGDATA grant IIS-1546482.

References

[1] E. Allender. Complexity theory lecture notes. `https://www.cs.rutgers.edu/~allender/lecture.notes/`, 1998.

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

[3] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[4] A. L. Blum and R. L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[5] S. Buss. Combinatorics: Circuit complexity. `http://www.math.ucsd.edu/~sbuss/CourseWeb/Math262A_2013F/`, 2013.

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

[7] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613. IEEE, 2013.

[8] B. DasGupta, H. T. Siegelmann, and E. Sontag. On the complexity of training neural networks with continuous activation functions. *IEEE Transactions on Neural Networks*, 6(6):1490–1504, 1995.

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

[10] S. Goel, V. Kanade, A. Klivans, and J. Thaler. Reliably learning the relu in polynomial time. *arXiv preprint arXiv:1611.10258*, 2016.

[11] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[12] B. D. Haeffele and R. Vidal. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*, 2015.

[13] J. Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. ACM, 1986.

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[15] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

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

[17] S. Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[18] D. M. Kane and R. Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. *arXiv preprint arXiv:1511.07860*, 2015.

[19] K. Kawaguchi. Deep learning without poor local minima. *arXiv preprint arXiv:1605.07110*, 2016.

[20] S. Kopparty. Topics in complexity theory and pseudorandomness. `http://www.math.rutgers.edu/~sk1233/courses/topics-S13/`, 2013.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Q. V. Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[23] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] J. Matousek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2002.

[25] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[26] R. Pascanu, G. Montufar, and Y. Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[27] A. A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

[28] B. Rossman, R. A. Servedio, and L.-Y. Tan. An average-case depth hierarchy theorem for boolean circuits. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1030–1048. IEEE, 2015.

[29] H. Royden and P. Fitzpatrick. *Real Analysis.* Prentice Hall, 2010.

[30] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 1, page 3, 2009.

[31] R. Saptharishi. A survey of lower bounds in arithmetic circuit complexity, 2014.

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. arXiv preprint arXiv:1312.6229, 2014.

[33] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[34] O. Shamir. Distribution-specific hardness of learning neural networks. *arXiv preprint arXiv:1609.01037*, 2016.

[35] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.

[36] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

[37] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[38] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[39] M. Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

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

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

[42] G. M. Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 1995.

[43] U. Zwick. Boolean circuit complexity. `http://www.cs.tau.ac.il/~zwick/scribe-boolean.html`, 1994.

A Auxiliary Lemmas

Now we will collect some straightforward observations that will be used often. The following operations preserve the property of being representable by a ReLU DNN.

Lemma A.1. [Function Composition] If $f_1 : \mathbb{R}^d \to \mathbb{R}^m$ is represented by a d, m ReLU DNN with depth $k_1 + 1$ and size s_1, and $f_2 : \mathbb{R}^m \to \mathbb{R}^n$ is represented by an m, n ReLU DNN with depth $k_2 + 1$ and size s_2, then $f_2 \circ f_1$ can be represented by a d, n ReLU DNN with depth $k_1 + k_2 + 1$ and size $s_1 + s_2$.

Proof. Follows from (1) and the fact that a composition of affine transformations is another affine transformation. □

Lemma A.2. [Function Addition] If $f_1 : \mathbb{R}^n \to \mathbb{R}^m$ is represented by a n, m ReLU DNN with depth $k + 1$ and size s_1, and $f_2 : \mathbb{R}^n \to \mathbb{R}^m$ is represented by a n, m ReLU DNN with depth $k + 1$ and size s_2, then $f_1 + f_2$ can be represented by a n, m ReLU DNN with depth $k + 1$ and size $s_1 + s_2$.

Proof. We simply put the two ReLU DNNs in parallel and combine the appropriate coordinates of the outputs. □

Lemma A.3. [Taking maximums/minimums] Let $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ be functions that can each be represented by $\mathbb{R}^n \to \mathbb{R}$ ReLU DNNs with depths $k_i + 1$ and size s_i, $i = 1, \ldots, m$. Then the function $f : \mathbb{R}^n \to \mathbb{R}$ defined as $f(\mathbf{x}) := \max\{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$ can be represented by a ReLU DNN of depth at most $\max\{k_1, \ldots, k_m\} + \log(m) + 1$ and size at most $s_1 + \ldots s_m + 4(2m - 1)$. Similarly, the function $g(\mathbf{x}) := \min\{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$ can be represented by a ReLU DNN of depth at most $\max\{k_1, \ldots, k_m\} + \lceil\log(m)\rceil + 1$ and size at most $s_1 + \ldots s_m + 4(2m - 1)$.

Proof. We prove this by induction on m. The base case $m = 1$ is trivial. For $m \geq 2$, consider $g_1 := \max\{f_1, \ldots, f_{\lfloor \frac{m}{2} \rfloor}\}$ and $g_2 := \max\{f_1, \ldots, f_{\lceil \frac{m}{2} \rceil}\}$. By the induction hypothesis (since $\lfloor \frac{m}{2} \rfloor, \lceil \frac{m}{2} \rceil < m$ when $m \geq 2$), g_1 and g_2 can be represented by ReLU DNNs of depths at most $\max\{k_1, \ldots, k_{\lfloor \frac{m}{2} \rfloor}\} + \lceil\log(\lfloor \frac{m}{2} \rfloor)\rceil + 1$ and $\max\{k_{\lceil \frac{m}{2} \rceil}, \ldots, k_m\} + \lceil\log(\lceil \frac{m}{2} \rceil)\rceil + 1$ respectively, and sizes at most $s_1 + \ldots s_{\lfloor \frac{m}{2} \rfloor} + 4(2\lfloor \frac{m}{2} \rfloor - 1)$ and $s_{\lceil \frac{m}{2} \rceil} + \ldots + s_m + 4(2\lfloor \frac{m}{2} \rfloor - 1)$, respectively. Therefore, the function $G : \mathbb{R}^n \to \mathbb{R}^2$ given by $G(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}))$ can be implemented by a ReLU DNN with depth at most $\max\{k_1, \ldots, k_m\} + \lceil\log(\lceil \frac{m}{2} \rceil)\rceil + 1$ and size at most $s_1 + \ldots + s_m + 4(2m - 2)$.

We now show how to represent the function $T : \mathbb{R}^2 \to \mathbb{R}$ defined as $T(x, y) = \max\{x, y\} = \frac{x+y}{2} + \frac{|x-y|}{2}$ by a 2-layer ReLU DNN with size 4 – see Figure 3. The result now follows from the fact that $f = T \circ G$ and Lemma A.1. □

Lemma A.4. Any affine transformation $T : \mathbb{R}^n \to \mathbb{R}^m$ is representable by a 2-layer ReLU DNN of size $2m$.

Proof. Simply use the fact that $T = (I \circ \sigma \circ T) + (-I \circ \sigma \circ (-T))$, and the right hand side can be represented by a 2-layer ReLU DNN of size $2m$ using Lemma A.2. □

Lemma A.5. Let $f : \mathbb{R} \to \mathbb{R}$ be a function represented by a $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k + 1$ and widths $w_1, \ldots, w_k$ of the k hidden layers. Then f is a PWL function with at most $2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k$ pieces.

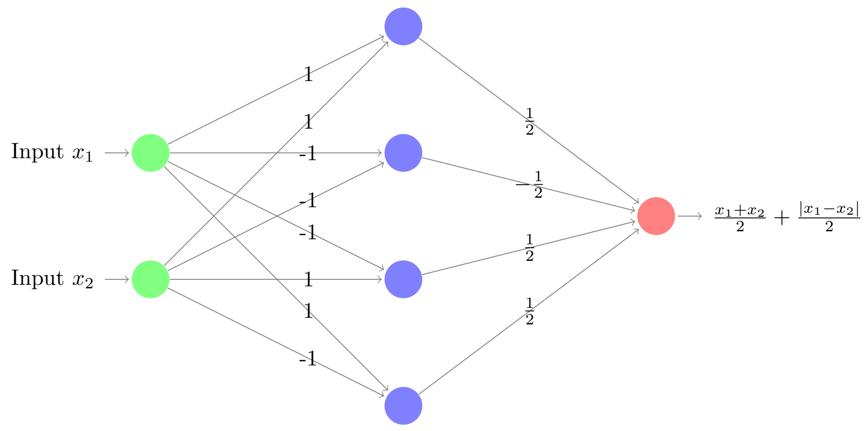


Figure 3: A 2-layer ReLU DNN computing $\max\{x_1, x_2\} = \frac{x_1+x_2}{2} + \frac{|x_1-x_2|}{2}$

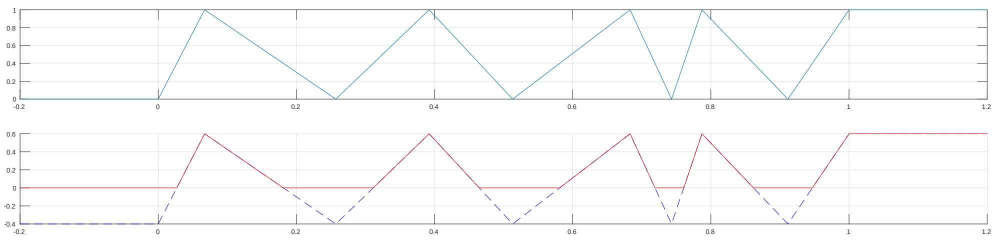


Figure 4: The number of pieces increasing after activation. If the blue function is f, then the red function $g = \max\{0, f + b\}$ has at most twice the number of pieces as f for any bias $b \in \mathbb{R}$.

Proof. We prove this by induction on k. The base case is $k = 1$, i.e, we have a 2-layer ReLU DNN. Since every activation node can produce at most one breakpoint in the piecewise linear function, we can get at most w_1 breakpoints, i.e., $w_1 + 1$ pieces.

Now for the induction step, assume that for some $k \geq 1$, any $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+1$ and widths $w_1, \ldots, w_k$ of the k hidden layers produces at most $2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k$ pieces.

Consider any $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+2$ and widths $w_1, \ldots, w_{k+1}$ of the $k+1$ hidden layers. Observe that the input to any node in the last layer is the output of a $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+1$ and widths $w_1, \ldots, w_k$. By the induction hypothesis, the input to this node in the last layer is a piecewise linear function f with at most $2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k$ pieces. When we apply the activation, the new function $g(x) = \max\{0, f(x)\}$, which is the output of this node, may have at most twice the number of pieces as f, because each original piece may be intersected by the x-axis; see Figure 4. Thus, after going through the layer, we take an affine combination of w_{k+1} functions, each with at most $2 \cdot (2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k)$ pieces. In all, we can therefore get at most $2 \cdot (2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k) \cdot w_{k+1}$ pieces, which is equal to $2^k \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k \cdot w_{k+1}$, and the induction step is completed. $\square$

Lemma A.5 has the following consequence about the depth and size tradeoffs for expressing functions with agiven number of pieces.

Lemma A.6. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a piecewise linear function with p pieces. If f is represented by a ReLU DNN with depth $k+1$, then it must have size at least $\frac{1}{2}kp^{1/k} - 1$. Conversely, any piecewise linear function f that represented by a ReLU DNN of depth $k+1$ and size at most s, can have at most $(\frac{2s}{k})^k$ pieces.

Proof. Let widths of the k hidden layers be $w_1, \ldots, w_k$. By Lemma A.5, we must have

$$2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k \geq p. \tag{12}$$

By the AM-GM inequality, minimizing the size $w_1 + w_2 + \ldots + w_k$ subject to (12), means setting $w_1 + 1 = w_2 = \ldots = w_k$. This implies that $w_1 + 1 = w_2 = \ldots = w_k \geq \frac{1}{2}p^{1/k}$. The first statement follows. The second statement follows using the AM-GM inequality again, this time with a restriction on $w_1 + w_2 + \ldots + w_k$. □