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 a ReLU DNN with one hidden layer to *global optimality*. This follows from our complete characterization of the ReLU DNN function class whereby we show that a $\mathbb{R}^n \to \mathbb{R}$ function is representable by a ReLU DNN *if and only if* it is a continuous piecewise linear function. The main tool used to prove this characterization is an elegant result from tropical geometry. Further, for the $n = 1$ case, we show that a single hidden layer suffices to express all piecewise linear functions, and we give tight bounds for the size of such a ReLU DNN. We follow up with gap results showing that there is a smoothly parameterized family of $\mathbb{R} \to \mathbb{R}$ "hard" functions that lead to an exponential blow-up in size, if the number of layers is decreased by a small amount. An example consequence of our gap theorem is that for every natural number N, there exists a function representable by a ReLU DNN with depth $N^2 + 1$ and total size N^3, such that any ReLU DNN with depth at most $N + 1$ will require at least $\frac{1}{2}N^{N+1} - 1$ total nodes.

Finally, we construct a family of $\mathbb{R}^n \to \mathbb{R}$ 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.

*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`

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 [12]. Furthermore, neural networks have finite VC dimension (depending polynomially on the network topology), 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 had mixed empirical success in early 1990s and were shown to be computationally hard to learn in the worst case [2]. Consequently, DNNs fell out of favor by late 90s.

Recently, there has been a resurgence of DNNs with the advent of deep learning [19]. 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 [11], 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 [10]. 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 [7, 10, 17, 18, 32]. 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 [31], but also considering alternate deep architectures such as convolutional neural networks [27], deep belief networks[11], and deep Boltzmann machines [25]. 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, [14, 28, 33, 34], as well as efforts which have tried to understand the non-convex nature of the training problem of DNNs better [8, 15]. 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 [1, 3, 5, 13, 16, 26, 29, 36] for various surveys of this deep and fascinating field. In particular, our gap results are inspired by results like the ones by Hastad [9], Razborov [23] and Smolensky [30] 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

We first set up the requisite notation to state our results formally.

2.1 Notation

Definition 1. For every $n \geq 1$, we define a function $\sigma_n : \mathbb{R}^n \to \mathbb{R}^n$ as follows

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

When the dimension n is clear from context, we will often drop the subscript and simply write $\sigma(x)$. σ_1 is called the *rectified linear unit* activation function in neural nets literature.

Definition 2. Let $U \subseteq \mathbb{R}^n$. We say $T\colon U \to \mathbb{R}^m$ is an *affine (or affine linear) transformation* over U if there exists a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{R}^m$ such that $T(\mathbf{u}) = A\mathbf{u} + \mathbf{b}$, for all $\mathbf{u} \in U$.

Definition 3. [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 $w_1, w_2, \ldots, w_k$ and a set of $k+1$ 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 $T_{k+1} : \mathbb{R}^{w_k} \to \mathbb{R}^{n_2}$. 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

$$f = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ T_2 \circ \sigma \circ T_1, \tag{2.1}$$

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 4 (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 closed sets whose union is $\mathbb{R}^n$, and f is affine linear over each set (note that the definition automatically implies continuity of the function). The number of pieces of f is the number of maximal connected subsets of $\mathbb{R}^n$ over which f is affine linear.

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

Definition 5. 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; we prove the converse.

Theorem 2.1. 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.

This uses a structural result from tropical geometry [21] about PWL functions. This gives an exact representation of the family of functions representable by ReLU DNNs. For $n = 1$ we can make the above statement stronger, with tight bounds on size, as follows:

Theorem 2.2. Given any piecewise linear function $\mathbb{R} \to \mathbb{R}$ with p pieces there exists a 2-layer DNN with p nodes that can represent f. Moreover, any 2-layer DNN that represents f has size at least $p-1$. Thus, any function represented by a ReLU DNN with more than 2 layers, can always be also represented by a 2-layer ReLU DNN.

Moreover, the above theorems imply that any function in $L^q(\mathbb{R}^n)$ can be arbitrarily well approximated by a ReLU DNN.

Theorem 2.3. Every function in $L^q(\mathbb{R}^n)$, $(1 \leq q \leq \infty)$ can be arbitrarily well-approximated by a ReLU DNN function. Moreover, for $n = 1$, any such $L^q(\mathbb{R}^n)$ 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.

2.2.2 Training 2-layer ReLU DNNs to global optimality

The *training problem* is the following: Given D data points $(x_i, y_i) \in \mathbb{R}^2$, $i = 1, \ldots, D$, 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 2.2 has the following consequence.

Theorem 2.4. There exists an algorithm to *optimally* train a 2-layer DNN of width w on D data points $(x_1, y_1), \ldots, (x_D, y_D) \in \mathbb{R}^2$, in time $O((2D)^w \text{poly}(D))$, i.e., the algorithm solves the following optimization problem to global optimality

$$\min \left\{ \sum_{i=1}^{D} \ell\big(\ T_2(\sigma(T_1(x_i))), y_i\ \big) : T_1 : \mathbb{R} \to \mathbb{R}^w, T_2 : \mathbb{R}^w \to \mathbb{R} \text{ are affine functions} \right\} \tag{2.2}$$

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

Key Insight: When the training problem is viewed as an optimization problem in the space of parameters of the ReLU DNN, it is a nonconvex, quadratic problem similar to a matrix factorization problem. However, because of the above equivalence proved by us between PWL functions and ReLU DNNs, **one can transform this nonconvex optimization problem to the space of slopes and intercepts of the function, where the problem becomes convex.**

Parallelizability: Our training algorithm is highly parallelizable. It solves $O((2D)^w)$ convex optimization problems and takes the minimum of all these values. Therefore, if one had $(2D)^w$ parallel processors, the TOTAL time taken to train would be roughly the time to do linear regression on a data set of size D, i.e., just poly(D).

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

We extend a result of Telgarsky [33, 34] about the depth-size trade-off for ReLU DNNs.

Theorem 2.5. 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 $\frac{1}{2} w^{\frac{k}{k'}} k' - 1$.

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

Theorem 2.6. For every $k \geq 1$, $w \geq 2$, every member of the family of hard functions in Theorem 2.5 has w^k pieces and the entire family is in one-to-one correspondence with the set

$$\bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}}. \tag{2.3}$$

The following is an immediate corollary of Theorem 2.5 by setting $k = N^{1+\epsilon}$ and $w = N$.

Corollary 2.7. For every $N \in \mathbb{N}$ 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 $(N^{1+\epsilon}) + 1$-layer DNN with size $N^{2+\epsilon}$, and if f is represented by a $N+1$-layer DNN, then this DNN must have size at least $\frac{1}{2} N \cdot N^{N^\epsilon} - 1$. Moreover, this family is in one-to-one correspondence with $\cup_{M>0} \Delta_M^{N^{2+\epsilon}-1}$.

A particularly illuminative special case is obtained by setting $\epsilon = 1$ in Corollary 2.7:

Corollary 2.8. For every natural number $N \in \mathbb{N}$, there is a family of functions parameterized by the set $\cup_{M>0} \Delta_M^{N^3-1}$ such that any f from this family can be represented by a N^2+1-layer DNN with N^3 nodes, and every $N+1$-layer DNN that represents f needs at least $\frac{1}{2} N^{N+1} - 1$ nodes.

We can also get hardness of approximation versions of Theorem 2.5 and Corollaries 2.7 and 2.8, with the same gaps (upto constant terms), using the following theorem.

Theorem 2.9. 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 $\mathcal{F}$ 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 $\text{comp}_{\mathcal{F}}(n,k,s)$ as the maximum number of pieces of a $\mathbb{R}^n \to \mathbb{R}$ function from $\mathcal{F}$ 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 [20, 22]. The best known families $\mathcal{F}$ are the ones from [20] and [33, 34] which achieve $\text{comp}_{\mathcal{F}}(n,k,s) = O(\frac{(s/k)^{kn}}{n^{n(k-1)}})$

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 2.10. 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 2.1 and 2.3 are proved in Section 3.1. Theorem 2.2 is proved in Section 3.2. Theorem 2.4 is in Section 4. Theorems 2.5 and 2.6 are proved in Section 5. Theorem 2.9 is proved in Section 6. Theorem 2.10 is proved in Section 7. Auxiliary results and structural lemmas appear in Appendix A.

2.4 Comparison with previous work

We are not aware of any previous work giving an exact characterization of the family of functions represented by any class of deep neural nets. Thus, results from Section 2.2.1 are unique in the literature on deep neural nets. Similarly, 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 is thought of as a fixed constant.

The results on depth-size trade-off from Section 2.2.3 have recent predecessors in Telgarsky's elegant theorems from [33, 34]. Our results extend Telgarsky's achievements in two ways:

(i) Telgarsky provides a *single* function for every $k \in N$ that can be represented by a k layer DNN with 2 nodes in each hidden layer, such that any ReLU DNN with $k' \leq k$ layers needs at least $2^{\frac{k-3}{k'}-1}k'$ nodes. This is also something characteristic of the "hard" functions in Boolean circuit complexity: they are usually a countable family of functions, not a "smooth" family of hard functions. In fact, in the last section of [33], 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 2.3).

(ii) As stated above, Telgarsky's family of hard functions is parameterized by a single natural number k and the gap in size is $2^{\frac{k-3}{k'}-1}k'$. In contrast, we show that for every *pair* of natural number w, k (and a point from the set in equation 2.3), there exists a "hard" function that proves a gap of $w^{\frac{k}{k'}}k'$. With the extra flexibility of choosing the parameter w, this gives stronger results with super exponential gaps, such as Corollaries 2.7 and 2.8.

We point out that Telgarsky's results in [34] 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 [34] are more general than our results in this paper, but with weaker gap guarantees.

Eldan-Shamir [28] 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 [24].

3 Expressing piecewise linear functions using ReLU DNNs

3.1 Every $\mathbb{R}^n \to \mathbb{R}$ PWL function is representable by a ReLU DNN

Proof of Theorem 2.1. Since σ_n is a continuous piecewise linear function for every $n \geq 1$, it is immediate from the expression 2.1 that every function represented by a ReLU DNN is a continuous piecewise linear, because composing continuous piecewise linear functions gives another piecewise linear function. So we simply have to show that every piecewise linear function from $\mathbb{R}^n \to \mathbb{R}$ is representable by a ReLU DNN.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a piecewise linear function with $\mathscr{P}$ as the corresponding polyhedral complex. For each cell $I \in \mathscr{P}$, let $g_I : \mathbb{R}^n \to \mathbb{R}$ be the affine function such that $f(\mathbf{x}) = g_I(\mathbf{x})$ for all $\mathbf{x} \in$ I.

Now by a theorem of Ovchinnikov [21, Theorem 2.1] from tropical geometry, there exists a finite set of subfamilies $S_1, \ldots, S_m \subseteq \mathscr{P}$ such that $S_1 \cup \ldots \cup S_m = \mathscr{P}$ and for every $\mathbf{x} \in \mathbb{R}^d$,

$$f(\mathbf{x}) = \max_{i=1,\ldots,m} \min_{I \in S_i} g_I(\mathbf{x}). \tag{3.1}$$

Each of these $g_I : \mathbb{R}^n \to \mathbb{R}$ is representable by a ReLU DNN by Lemma A.4. Then using Lemma A.3 on the expression (3.1) gives the result. □

Proof of theorem 2.3. Knowing that any piecewise linear function $\mathbb{R}^n \to \mathbb{R}$ is representable by a ReLU DNN, 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 $1 \leq p \leq \infty$. □

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

Proof of Theorem 2.2. 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

$$f(x) = \begin{cases} 0 & x \leq a \\ r(x-a) & x > a \end{cases} \tag{3.2}$$

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,

$$g(x) = \begin{cases} t(x-a) & x \leq a \\ 0 & x > a \end{cases} \tag{3.3}$$

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 (3.2) and (3.3), 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 (3.2) with slope r and breakpoint $a = a_{m-1}$, and $m-1$ functions $g_1, \ldots, g_{m-1}$ of the form (3.3) 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 $r, t_1, \ldots, t_{m-1}$:

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

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. $\square$

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 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 $p - 1$.

4 Training 2-layer ReLU DNNs optimally

Proof of Theorem 2.4. 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 2.2, to solve problem (2.2) 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 (2.2) is equivalent to the problem

$$\min \left\{ \sum_{i=1}^{D} \ell(f(x_i), y_i) : f \text{ is piecewise linear with } w \text{ pieces} \right\}. \tag{4.1}$$

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 $(a_1, b_1), \ldots, (a_2, b_2), \ldots, (a_w, b_w)$:

$$S_j(b_{j+1} - b_j - (a_j - a_{j+1})x_{i_j}) \geq 0 \tag{4.2}$$

$$S_j(b_{j+1} - b_j - (a_j - a_{j+1})x_{i_j+1}) \leq 0 \tag{4.3}$$

$$S_j(a_{j+1} - a_j) \geq 0 \tag{4.4}$$

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

$$\min\{\sum_{i=1}^{D} \ell(f(x_i) - y_i) : f \in \mathrm{PWL}^1_{I,S}\} = \min\{\sum_{j=1}^{w} \left(\sum_{i \in D_j} \ell(a_j \cdot x_i + b_j - y_i) \right) : (a_j, b_j) \text{ satisfy } (4.2)\} \tag{4.5}$$

The right hand side of the above equation is the problem of minimizing a convex objective subject to linear constraints. Now, to solve (4.1), we need to simply solve the problem (4.5) 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 $O(\mathrm{poly}(D))$. Therefore, the total running time is $O((2D)^w \mathrm{poly}(D))$. □

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

Definition 6. Let $M > 0$ be a given real number, and let $p \in \mathbb{N}$. Let $\mathbf{a} \in \Delta^p_M$. We define a function $h_{\mathbf{a}} : \mathbb{R} \to \mathbb{R}$ which is piecewise linear over the segments $(-\infty, 0], [0, \mathbf{a}_1], [\mathbf{a}_1, \mathbf{a}_2], \ldots, [\mathbf{a}_p, 1], [1, +\infty)$ 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 $[\mathbf{a}_p, 1]$. 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 $\mathbf{a}^1, \ldots, \mathbf{a}^k \in \Delta^p_M$, 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. □

Proof of Theorem 2.6. Given $k \geq 1$ and $w \geq 2$, choose any point

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

By Definition 6, 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. □

Proof of Theorem 2.5. Follows from Theorem 2.6 and Lemma A.6. □

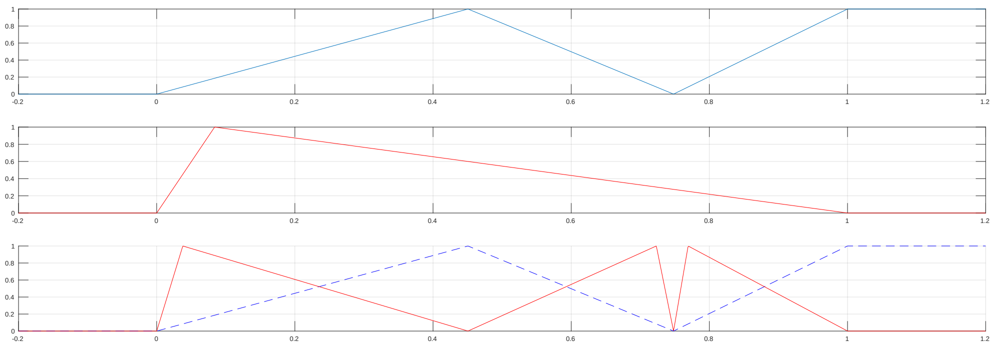


Figure 1: Top: $h_{\mathbf{a}^1}$ with $\mathbf{a}^1 \in \Delta_1^2$ with 3 pieces in the range $[0,1]$. Middle: $h_{\mathbf{a}^2}$ with $\mathbf{a}^2 \in \Delta_1^1$ with 2 pieces in the range $[0,1]$. Bottom: $H_{\mathbf{a}^1,\mathbf{a}^2} = h_{\mathbf{a}^2} \circ h_{\mathbf{a}^1}$ with $2 \cdot 3 = 6$ 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.

6 L^1-norm gap analysis

Proof of Theorem 2.9. 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. $\square$

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

Definition 7. 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

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

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 [35].

Theorem 7.1. The following are both true.

1. $|\operatorname{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))| \le (m-1)^{n-1}$. 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 8. The set $S(n, m)$ will denote the set of $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in R^n \times \ldots \times \mathbb{R}^n$ such that $|\operatorname{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))| = (m-1)^{d-1}$.

Lemma 7.2. 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 7.1 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 that

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

$\square$

Proposition 7.3. Given any tuple

$$(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in S(n, m)$$

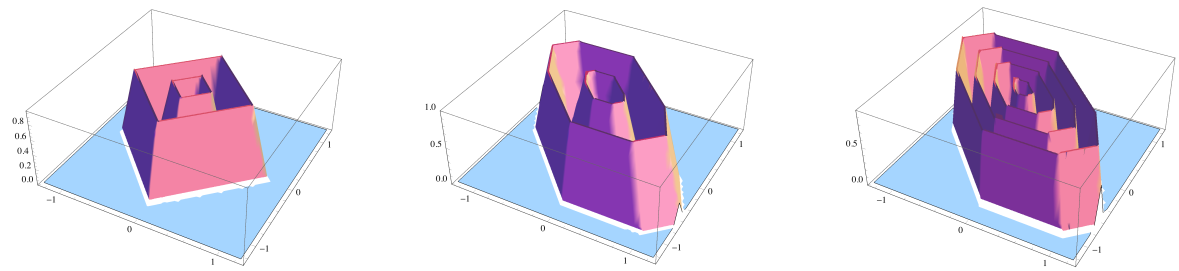
and any point

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

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 $2m + wk$.



(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.)

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 7.2 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 7.1, and $H_{\mathbf{a}^1,\ldots,\mathbf{a}^k}$ has w^k pieces by Lemma 5.1. □

Proof of Theorem 2.10. Follows from Proposition 7.3. □

8 Discussion

Deep neural networks are a popular function class for many machine learning problems. DNNs based on sigmoidal activation and arbitrary non-constant bounded activation functions have been known to be universal approximators for $Ł^q(\mathbb{R}^n)$. Here, we show that ReLU DNNs, i.e. those based on non-saturating nonlinearities, can also approximate any function in $Ł^q(\mathbb{R}^n)$ arbitrarily well. Furthermore, we show that the function class represented by ReLU DNNs is exactly that of all piecewise linear functions.

A second main contribution of the paper is to show that ReLU DNNs with a single hidden layer can be learned efficiently. That is, given a finite sample of input-output pairs from an unknown ReLU DNN, we give an algorithm that can recover the ReLU DNN exactly, in time that is polynomial in the sample size. This is a first of its kind result as neural networks are known to be computationally hard to train. In fact, it is well known that implementing the empirical risk minimization (ERM) rule (essentially the training problem we describe in the paper) is NP hard even for a single-input single-output DNN with a single hidden layer with three nodes [4]. However, note that the existing hardness results are for threshold activation functions and hence they do not contradict our results. Thus, we provide evidence that restricting to rectified linear units helps to avoid such computational complexity barriers.

Finally, we precisely specify gaps in the function classes represented by ReLU DNNs at different depths with constrained sizes, and give the first ever construction of a *continuum* of hard functions for different depths and sizes. For $\mathbb{R} \to \mathbb{R}$ ReLU DNNs this leads us to improve on the previously known super-polynomial lower bounds on circuit sizes. For $\mathbb{R}^n \to \mathbb{R}$ circuits we are able to construct more complex functions than previously known.

This work suggests several interesting future research directions. While the proposed algorithm depends only polynomially on the sample size, it depends exponentially on the size of the network. It is possible that a more efficient algorithm exists that has a milder dependence on the network size. Empirical results suggest that a simple stochastic gradient descent algorithm works well for ReLU DNNs. It also remains to be seen how the proposed training algorithm can be extended to ReLU DNNs with n inputs. Finally, for $\mathbb{R}^n \to \mathbb{R}$ for $n > 1$ we can currently give superpolynomial gaps between superlinearly separated depths but ideally we would like to get optimal gaps between consecutive constant depths or between logarithmic and constant depths.

A Auxiliary Lemmas

In this short appendix, 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 (2.1) 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 $k_1 + 1$ and size s_1, and $f_2 : \mathbb{R}^n \to \mathbb{R}^m$ is represented by a n, m ReLU DNN with depth $k_2 + 1$ and size s_2, then $f_1 + f_2$ can be represented by a d, m ReLU DNN with depth $\max\{k_1, k_2\} + 1$ and size $s_1 + s_2$.

Proof. 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. $\square$

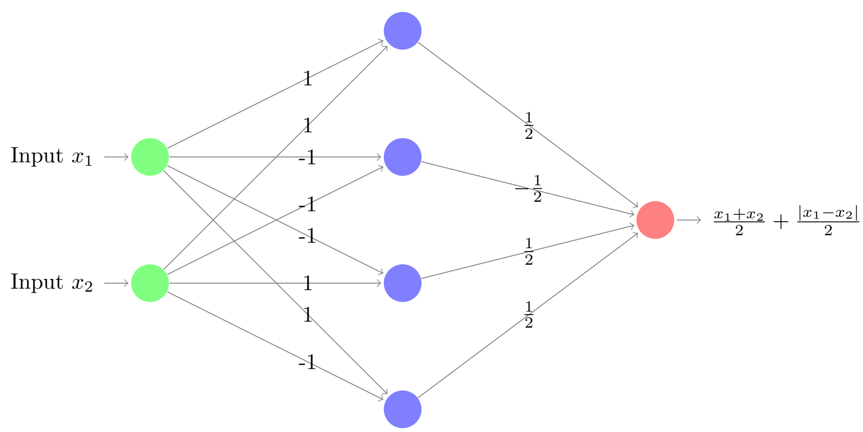


Figure 3: A 2-layer ReLU DNN computing $\max\{x_1, x_2\} = \frac{x_1+x_2}{2} + \frac{|x_1-x_2|}{2}$

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. $\square$

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.

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

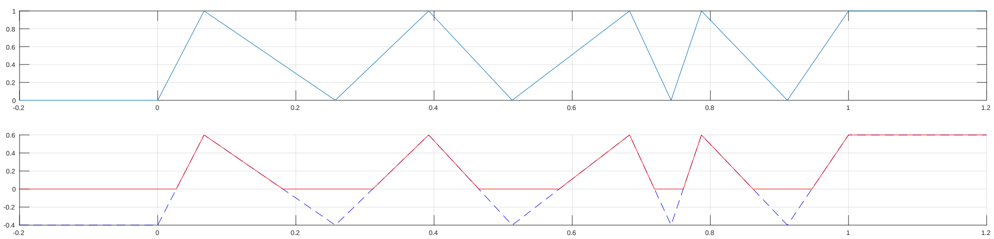


Figure 4: The number of pieces increasing after activation. If the blue function is f, then the red function $g = \max\{0, f + b\}$ has at most twice the number of pieces as f for any bias $b \in \mathbb{R}$.

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

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

By the AM-GM inequality, minimizing the size $w_1 + w_2 + \ldots + w_k$ subject to (A.1), 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 $w_1 + w_2 + \ldots + w_k$. $\square$

References

[1] Eric Allender. Complexity theory lecture notes. https://www.cs.rutgers.edu/~allender/lecture.notes/.

[2] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations.* cambridge university press, 2009.

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

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

[5] Sam Buss. Combinatorics: Circuit complexity. http://www.math.ucsd.edu/~sbuss/CourseWeb/Math262A_2013F/.

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

[7] George E Dahl, Tara N Sainath, and Geoffrey 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] Benjamin D Haeffele and René Vidal. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*, 2015.

[9] Johan 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.

[10] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara 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.

[11] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

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

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

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

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

[16] Swastik Kopparty. Topics in complexity theory and pseudorandomness. http://www.math.rutgers.edu/~sk1233/courses/topics-S13/.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Quoc 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.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[20] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[21] Sergei Ovchinnikov. Max-min representation of piecewise linear functions. *Contributions to Algebra and Geometry*, 43(1):297–302, 2002.

[22] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[23] Alexander 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.

[24] Benjamin Rossman, Rocco A Servedio, and Li-Yang 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.

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

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

[27] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. arXiv preprint arXiv:1312.6229, 2014.

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

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

[30] Roman 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.

[31] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

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

[34] Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

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

[36] Uri Zwick. Boolean circuit complexity. http://www.cs.tau.ac.il/~zwick/scribe-boolean.html.