Obstacles to Depth Compression of Neural Networks

Will Burstein and John Wilmes

Brandeis University, Waltham, Massachusetts, USA
{wburstein, wilmes}@brandeis.edu

Abstract. Massive neural network models are often preferred over smaller models for their more favorable optimization landscapes during training. However, since the cost of evaluating a model grows with the size, it is desirable to obtain an equivalent compressed neural network model before deploying it for prediction. The best-studied tools for compressing neural networks obtain models with broadly similar architectures, including the depth of the model. No guarantees have been available for obtaining compressed models with substantially reduced depth. In this paper, we present fundamental obstacles to any algorithm achieving depth compression of neural networks. In particular, we show that depth compression is as hard as learning the input distribution, ruling out guarantees for most existing approaches. Furthermore, even when the input distribution is of a known, simple form, we show that there are no *local* algorithms for depth compression.

1 Introduction

Perhaps the clearest trend of the past decade of deep learning has been the ever-increasing size of neural network (NN) models, fueled by advances in hardware acceleration, larger datasets, and improved optimization techniques.

It may be practical to train models on datacenter scale computational resources, but it is still desirable to obtain models that can be deployed to more modest hardware for inference tasks. In particular, for inference tasks on individual user data where low latency is important, deploying models onto smartphones or similarly constrained hardware may be necessary. Even in cases where inference tasks can be performed within data centers, the cost of evaluating a model, including its environmental impact, may still be significant if many real-time inference requests must be satisfied.

While small models are desirable at the time of inference, there are equally good reasons at training time for the rapid growth in scale of neural network models. Although depth-3 neural networks are already "universal models" in terms of their representational power, much deeper networks are empirically observed to have desirable optimization characteristics. In practice, good generalization performance is often easier to elicit from massive, very deep neural network models.

In order to combine the benefits of massively overparameterized models at training time with the computational efficiency of smaller models for inference, it is therefore desirable to convert large models to smaller ones before deployment. This is the task of model compression.

A classic approach to model compression task is pruning, i.e., zeroing out entries in the weight matrices of a NN. Some of the earliest work on this idea used information about second derivatives of the NN to find appropriate weights to trim [23,17]. A simpler idea is to simply drop all entries in the weight matrices below some threshold, and then retrain the network (keeping all dropped entries zero) [16]. A more sophisticated approach is to incorporate the pruning into the training [31]. A second but conceptually related approach to model compression is via low-rank factorization. Following the observation that many neural networks trained on real-world data have approximately low-rank weight matrices [8], several works have sought to compress NNs by replacing weight matrices with low-rank approximations [19,9,29].

Both the pruning and low-rank factorization approaches preserve the overall network architecture while reducing the complexity of the weight matrices, as measured in terms of either their sparsity or rank. These techniques can therefore reduce the number of parameters in the model, providing some improvement in the cost of inference, but they leave the depth of the network unchanged. Unfortuantely, the depth of a network is a crucial hyperparameter for determining the cost of inference. Evaluation of a single layer of a neural network is amenable to parallelization, and graphics hardware is particularly good at accelerating this computation. By contrast, evaluating a deep neural network is inherently sequential.

Thus, algorithms for neural network model compression that substantially reduce depth are highly desirable. Unfortunately, depth compression has been much less studied than pruning or low-rank compression techniques. The workhorse technique is a specialization of the "student–teacher" learning framework. In this approach, a pretrained "teacher" network is used to train a "student" network with an entirely different architecture [4,1]. The teacher network gives access to finer features than the ultimate network output, such as the relative certainties the teacher network assigns to possible outputs [18], or the intermediate-layer representations [25]. However, provable guarantees for the accuracy of student networks relative to their teachers remain elusive.

The goal of this paper is to initiate the study of provable guarantees for depth compression. We obtain the first nontrivial lower bounds for the problem, illustrating two fundamental obstacles to any depth compression algorithm. First, we give a general reduction from the problem of depth compression to the problem of learning the input distribution. This allows us to lift distribution-learning lower bounds to the setting of depth compression. Second, even when the input distribution is known and well-behaved, we rule out a natural class of divide-and-conquer approaches for depth compression, showing that "local" algorithms cannot in general achieve any nontrivial compression guarantees. Finally, we

conduct experiments illustrating how existing depth compression techniques fail in general.

Our ultimate goal is for positive algorithmic guarantees, and the present results are motivated by the philosophy that lower bounds can serve as a guideposts for what may be algorithmically feasible. Indeed, our lower bounds and experiments together outline a set of plausible assumptions under which provable guarantees for depth compression may be achieved (see Section 5).

2 Learning and Compression

Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be a map, and let $\mathcal{H}$ be a family of maps $h : \mathbb{R}^n \to \mathbb{R}^k$ (the hypothesis space). An $(\varepsilon, \mathcal{H})$-*compression* of f over a distribution D on $\mathbb{R}^n$ is a a map $h \in \mathcal{H}$ which is ε-close to f in mean-squared error, i.e., such that $\mathbb{E}_{x\sim D}\|f(x) - h(x)\|^2 < \varepsilon$.[1] Let $\mathcal{D}$ be a family of distributions on $\mathbb{R}^n$. A (randomized) algorithm for the $(f, \mathcal{H}, \mathcal{D})$ **compression problem** takes as input the explicit neural network representation of f, and some number m of samples from a fixed but unknown distribution $D \in \mathcal{D}$, and produces a map $h : \mathbb{R}^n \to \mathbb{R}^k$, with the guarantee that for any $\varepsilon > 0$ if $m \geq \text{poly}(1/\varepsilon, 1/\delta, \text{size}(f))$ then h is an $(\varepsilon, \mathcal{H})$-compression of f over D. where h is a random variable of the samples and the algorithm's internal randomness. The algorithm is *efficient* if it runs in time $\text{poly}(1/\varepsilon, 1/\delta, \text{size}(f))$. We will study $(f, \mathcal{H}, \mathcal{D})$ compression problems where f is an explicit deep neural network model and $\mathcal{H}$ is a family of much shallower neural networks.

Thus, the compression problem is similar to a PAC learning problem, with the additional explicit input of an improper[2] representation f of the target concept, and with the distribution-free assumption relaxed. There are well known examples where proper learning is NP-hard, although improper learning is tractable—for example, 3-term DNFs can be efficiently learned improperly as 3-term CNFs, but are NP-hard to learn properly. Here, we consider a different problem: both proper and improper learning may be hard, but transforming an improper model into a proper model (the compression problem) may still be tractable. As a simple example, learning noisy parities from random samples is widely believed to be hard, but with query access (as from some improper representation), the parity can be recovered using the Goldreich-Levin algorithm. The additional input of an improper representation is therefore essential to the complexity of the compression problem, and hardness results for learning do not directly transfer to the compression setting. Nevertheless, we will show in this paper how hardness results for learning can be used to obtain hardness results for neural network depth compression problems.

[1] In typical applications, f itself will be an approximation of some concept g known only through labeled examples, and the real goal is find an approximation of g in $\mathcal{H}$. To simplify our discussion, we will not attempt to find a compression of f which appoximates this concept g better than f does itself, and so g can be safely ignored.

[2] "Improper" in the sense of not belonging to the hypothesis class $\mathcal{H}$.

In order to have a sensible compression problem, in addition to bounding the depth of networks in $\mathcal{H}$, we must insist that their *total size* be bounded by some polynomial in the size of f. The most obvious measure of neural network size is perhaps the total number of neurons. However, we will instead use a finer-grained measurement that accounts for the magnitude of the weights in the network:

Definition 1. *Let $d \geq 1$, $\Lambda > 0$. Define $\mathrm{NN}_{\Lambda,d}$ to be the set of d-layer neural networks using 1-Lipschitz activation functions and with each weight matrix W having Frobenius norm bounded by Λ. That is, $f : \mathbb{R}^n \to \mathbb{R}^k$ belongs to $\mathrm{NN}_{\Lambda,d}$ if there exist integers $\ell_0, \ldots, \ell_d$ with $\ell_0 = n$ and $\ell_d = k$, and maps $g_i : \mathbb{R}^{\ell_{i-1}} \to \mathbb{R}^{\ell_i}$ for $1 \leq i \leq d$ of the form $g_i(x) = \sigma(Wx + b)$, where $W \in \mathbb{R}^{\ell_i \times \ell_{i-1}}$ satisfies $\|W\|_F \leq \Lambda$, and $b \in \mathbb{R}^{\ell_i}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a 1-Lipschitz map applied componentwise, such that*

$$f = g_d \circ g_{d-1} \circ \cdots \circ g_1 .$$

The maps g_i are called the layers of f.

For example, a neural network $f : \mathbb{R}^n \to \mathbb{R}$ with ReLU activations, a single hidden layer of m neurons, and bounded entries in its weight matrices, would belong to $\mathrm{NN}_{O(mn),2}$.

We will give lower bounds for compression problems conditioned on well-known complexity hypotheses, as well as unconditionally in the statistical query framework, which we now review. For computational problems over a distribution D (e.g., supervised learning of a concept), classical algorithms receive as input some number of samples from D. Often, these samples are used only to estimate the means of various random variables over the distribution D: for example, training a neural network by gradient descent requires labeled examples only in order to estimate the expected gradient at various points in parameter space. When an algorithm is formulated so that it does not require any access to D other than to query the expected value of bounded random variables over the distribution, it is called a *statistical query (SQ)* algorithm. The study of SQ algorithms was initiated be Kearns in 1993 [20], and has become an extremely powerful tool for algorithmic analysis [14,2,10,6]. The vast majority of computational problems known to admit efficient algorithms in fact admit efficient SQ algorithms. Unconditional lower bounds for SQ algorithms are also available, in particular characterizing the complexity of learning neural network models [30,27].

Formally, given some distribution D over a set X, let $\mathrm{vSTAT}_D(\tau)$ be an oracle that, when presented with query $\phi : X \to [0, 1]$, returns a value v satisfying $|\sqrt{v} - \sqrt{\mathrm{E}_{x\sim D}(\phi(x))}| < \tau$ [12]. A statistical query algorithm for a computational problem over a distribution D is a randomized algorithm that accesses D only via queries to an oracle $\mathrm{vSTAT}_D(\tau)$. Simulating $\mathrm{vSTAT}_D(\tau)$ by estimating $\mathrm{E}_{x\sim D}(\phi(x))$ from samples in general requires $\Omega(1/\tau)$ samples from D. We therefore define the *total complexity* of a statistical query algorithm using the $\mathrm{vSTAT}_D(\tau)$ oracle to be $\max\{1/\tau, d\}$ where d is the number of queries performed by the algorithm.

Theorem 1. *For any $\Lambda, d > 2$ there exists $f \in \mathrm{NN}_{\Lambda,d}$ and a family $\mathcal{D}$ of distributions such that*

1. *for every $D \in \mathcal{D}$, there exists a (lossless) $(0, \mathrm{NN}_{\Lambda,3})$-compression of f over D,*
2. *but the total complexity of $(f, \mathrm{NN}_{\Lambda,d-1}, \mathcal{D})$-compression is $\exp(\Omega(\Lambda))$.*

Similar statements can be made for general (not necessarily SQ) algorithms, under some reasonable complexity hypotheses. For example, in the Learning Parities with Noise (LPN) problem, a learning algorithm is given access to examples $x \in \{\pm 1\}^n$ drawn from the uniform distribution on the hypercube, and labeled according to some unknown parity function $h : \{\pm 1\}^n \to \{\pm 1\}$, with these labels randomly flipped with noise rate η. The algorithm's task is to find a function which is ε-close to h. The problem is notoriously difficult and its intractability has been frequently assumed [3,13,21,22].

LPN Hypothesis. For any constants $0 < \eta, \varepsilon < 1/2$, there is no $\mathrm{poly}(n)$-time algorithm solving the LPN problem with noise rate η to accuracy ε.

Theorem 2. *For any $\varepsilon > 0$ and $\Lambda, d > 2$ there exists $f \in \mathrm{NN}_{\Lambda,d}$ and a family $\mathcal{D}$ of distributions such that, under the LPN hypothesis,*

1. *for every $D \in \mathcal{D}$, there exists a $(\varepsilon, \mathrm{NN}_{\Lambda,3})$-compression of f over D,*
2. *but $(f, \mathrm{NN}_{\Lambda,d-1}, \mathcal{D})$-compression does not admit a polynomial-time algorithm.*

The theorems above show that regardless of how the improper representation f is used, compression is hard unless the input distribution is known. When the distribution is known, compression algorithms that achieve guarantees beyond those available for standard supervised learning problems must rely nontrivially on the improper representation f.

Existing algorithms for NN depth compression make quite coarse use of the improper representation. One of the foundational works on model compression [4] proposes to compress massive ensemble models to smaller NN models simply by training the smaller model by gradient descent on data labeled by the ensemble model (along with a simple method for augmenting the set of unlabeled data in the case when insufficient unlabeled data is available). The same approach has empirically seen some success for compressing deep NN models to shallower architectures [1]. The most successful family of techniques has been the *knowledge distillation* approach of [18]. In its original formulation for classification problems, the deep representation f is assumed to have a final softmax layer; rather than training a shallow student network directly on labels generated from the outputs of f, the student is trained using mean squared error on the penultimate layer of f, representing the relative certainties f assigns to each category. This approach was extended in other works to train student models with different architectures on intermediate-layer features from the original model f [25], and beyond the classification setting [5].

None of these algorithms can succeed at compression without strong assumptions on the form of the improper representation f, and in particular they require

f to have opaque regularization properties. Specifically, knowledge distillation and its cousins are empirically observed to work well for improper representations f obtained by trainining a deep neural network using gradient descent; without the "implicit bias" [24,15,28] imposed by the training of f, there is no reason to expect algorithmic guarantees for compression via knowledge distillation beyond the guarantees available for general learning problems.

Compression algorithms that work without strong assumptions on the regularization of f must instead make use of the detailed representation of f provided as input. This will not be an easy task. In the following Theorem 3, we will rule out natural divide-and-conquer approaches to making use of the improper representation f.

Let $f \in \mathrm{NN}_{\Lambda,d}$, and let $g_1, \ldots, g_d$ be the layers of f. For $1 \leq i \leq j \leq d$, the *slice* $f^{(i:j)}$ of f is the map $g_j \circ g_{j-1} \circ \cdots \circ g_i$. Given a distribution D on the input space of f, the distribution induced by D on the ith layer of f is the distribution of the random variable $f^{[1:i]}(x)$ over $x \sim D$.

Definition 2 (Locally compressible). *Let $f \in \mathrm{NN}_{\Lambda,d}$ and let D be a distribution on the input space of f. We say f is (ε, s, t)-locally compressible if for some $1 \leq i \leq d - t$ and $\Lambda' = \mathrm{poly}(\Lambda, d)$, there exists an $(\varepsilon, \mathrm{NN}_{\Lambda',s})$-compression of $f^{[i:i+t]}$ over the distribution induced by D on the ith layer of f.*

A natural divide-and-conquer approach for depth compression of a neural network would take some number of consecutive layers of the network, and replace those layers with a shallower (but perhaps wider) approximation. By iterating such an approach until no slice can be compressed, we might obtain a much shallower network. In the following theorem, we rule out such local algorithms by observing that there are arbitrarily deep networks admitting lossless (global) compressions, but which are locally incompressible.

Theorem 3. *There is some $c > 0$ such that for any $\Lambda, d > 3$, there is a neural network $f \in \mathrm{NN}_{\Lambda,d}$ and an input distribution D such that*

1. *f is not $(c, 2, 3)$-locally compressible over D,*
2. *but there exists a (lossless) $(0, \mathrm{NN}_{\Lambda(d-3),3})$-compression of f over D.*

3 Compression vs. Distribution Learning

Let D_0 be a probability distribution on $\mathbb{R}^n$. By a *family of η-perturbations of D_0* we mean a family $\mathcal{D}$ of distributions D given by random variables of the form $x + \eta s(x, D)v$, where $x \sim D_0$, $v \in \mathbb{R}^n$ is a fixed unit vector, and $s(x, D) \in \{\pm 1\}$ is a (deterministic) function of x and D, which for fixed D is measurable as a function of x with respect to D_0. Given such a family $\mathcal{D}$, we say a map $f : \mathbb{R}^n \to \mathbb{R}^k$ is a *Δ-separator* for $\mathcal{D}$ if it is measurable with respect to every distribution in $\mathcal{D}$ and $\|f(x + \eta v) - f(x - \eta v)\| \geq \Delta$ for all $x \in \mathrm{supp}(D_0)$.

A simple induction argument gives the following:

Proposition 1. *If $f \in \mathrm{NN}_{\Lambda,d}$ then f is Λ^d-Lipschitz.*

The following is the main technical lemma used in the proofs of Theorems 1 and 2. We denote by $\mathrm{D_{TV}}(D_1, D_2)$ the total variation distance between the distributions D_1 and D_2.

Lemma 1. *Let $\mathcal{D}$ be a family of η-perturbations of a distribution D_0 on $\mathbb{R}^n$, let f be a Δ-separator for $\mathcal{D}$, and let g be an $(\varepsilon, \mathrm{NN}_{\Lambda,d})$-compression of f over some (unknown) distribution $D \in \mathcal{D}$. Suppose $\eta\Lambda^d \leq \Delta/4$. There is an η-perturbation $\tilde{D}$ of D_0 satisfying $\mathrm{D_{TV}}(\tilde{D}, D) = 16\varepsilon/\Delta^2$ and an efficient algorithm that, given query access to f and g:*

1. *produces samples from $\tilde{D}$, given access to samples from D_0;*
2. *computes the probability density of points in $\mathbb{R}^n$ under $\tilde{D}$, given query access to the probability density of points under D_0.*

Proof. Let $v \in \mathbb{R}^n$ be the vector and $s : \mathbb{R}^n \times \mathcal{D} \to \{\pm 1\}$ the map characterizing $\mathcal{D}$ as a family of η-perturbations of D_0. Let $D \in \mathcal{D}$ and let $g \in \mathrm{NN}_{\Lambda,d}$ be a compression of f. Define the following map z on the support of D_0:

$$z(x_0) = \begin{cases} x + \eta v & \|g(x_0) - f(x_0 + \eta v)\| < \|g(x_0) - f(x_0 - \eta v)\| \\ x - \eta v & \text{otherwise} \end{cases}.$$

Let $\tilde{D}$ denote the distribution of $z(x_0)$, where $x_0 \sim D_0$.

We now argue that $\tilde{D}$ is close to D in total variation distance. Fix x in the support of D and let $x_0 = x - \eta s(x, D)v$. We have by Proposition 1 and the triangle inequality that

$$\begin{aligned} \|g(x_0) - f(x_0 + \eta s(x, D)v)\| &\leq \|g(x_0) - g(x)\| + \|g(x) - f(x)\| \\ &\leq \eta\Lambda^d + \|g(x) - f(x)\| . \end{aligned}$$

Let $\theta = (\Delta/2 - \eta\Lambda^d)$ and suppose further that $\|g(x) - f(x)\| < \theta$, so that $\|g(x_0) - f(x)\| < \Delta/2$. In this case, by the triangle inequality and the fact that f is a Δ-separator,

$$\begin{aligned} \|g(x_0) - f(x_0 - \eta s(x, D)v)\| &> \|f(x_0 + \eta v) - f(x_0 - \eta v)\| - \|g(x_0) - f(x)\| \\ &> \Delta/2 > \|g(x_0) - f(x)\| . \end{aligned}$$

So if $\|g(x) - f(x)\| < \theta$, we have $z(x_0) = x_0 + \eta s(x, D)v$. Hence,

$$\begin{aligned} \mathrm{D_{TV}}(D, \tilde{D}) &\leq \Pr_{x_0 \sim D_0}(z(x_0) \neq x_0 + \eta s(x_0, D)v) \\ &\leq \Pr_{x \sim D}(\|g(x) - f(x)\| \geq \theta) \end{aligned}$$

so it suffices to bound this latter probability. We have by assumption that

$$\mathrm{E}_{x \sim D}(\|g(x) - f(x)\|^2) < \varepsilon .$$

So by Markov's inequality,

$$\Pr_{x \sim D}(\|g(x) - f(x)\| \geq \theta) = \Pr_{x \sim D}(\|g(x) - f(x)\|^2 \geq \theta^2) < \varepsilon/\theta^2 \leq 16\varepsilon/\Delta^2 ,$$

as desired.

It remains only to observe that $\tilde{D}$ admits an efficient sampling algorithm and probability density computation algorithm, given corresponding access to D_0 and query access to f and g. Sampling from $\tilde{D}$ is the same as sampling x_0 from D_0 and computing $z(x_0)$, which requires one query of g and two of f. To compute probability densities $\tilde{p}$ under $\tilde{D}$, given probability densities p for D_0, we compute the density $\tilde{p}(x)$ at a point x as

$$\tilde{p}(x) = p(x - \eta v)\mathbf{1}(z(x - \eta v) = x) + p(x + \eta v)\mathbf{1}(z(x + \eta v) = x)\,.$$

We denote by $\mathcal{P}_n$ the set of parity functions $h : \{\pm 1\}^n \to \{\pm 1\}$ on the n-dimensional Boolean hypercube. We recall two standard results concerning parities.

Lemma 2. *A parity function $h \in \mathcal{P}_n$ can be represented exactly by a neural network in $\mathrm{NN}(1, 3)$ with $O(n)$ gates.*

Because the parities are pairwise uncorrelated, the set of parity functions has large "statistical dimension," from which it follows by a standard argument that the total complexity of any statistical algorithm for learning parities is also large. See, e.g., [12].

Lemma 3. *The total complexity of learning parities in $\mathcal{P}_n$ over the uniform distribution on the hypercube to within accuracy $\geq 3/4$ is $\exp(\Omega(n))$.*

Proof (Proof of Theorem 1). The theorem follows by applying Lemma 1 to an appropriate choice of map $f \in \mathrm{NN}_{\Lambda,d}$ and family $\mathcal{D}$ of distributions.

For some n to be defined, let $f : \mathbb{R}^{n+1} \to \mathbb{R}$ be given by the layer-d neural network with first layer

$$f_1(x_1, \ldots, x_{n+1}) = \Lambda x_{n+1}$$

and $f_i : \mathbb{R} \to \mathbb{R}$ given by $f_i(x) = \Lambda x$ for all $1 < i \leq d$. Clearly $f \in \mathrm{NN}_{\Lambda,d}$.

We now define $\mathcal{D}$ as a family of η-pertubations of a distribution. Let $\pi : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be the projection onto the first n coordinates, $\pi(x_1, \ldots, x_{n+1}) = (x_1, \ldots, x_n)$. Let X be the embedding of the n-dimensional hypercube in $\mathbb{R}^{n+1}$ given by

$$X = \{x = (x_1, \ldots, x_n, 0) : \pi(x) \in\in \{\pm 1\}\}$$

and let D_0 be the uniform distribution on X. Let v be the standard basis vector $(0, ..., 0, 1) \in \mathbb{R}^{n+1}$. Let $\eta = \Lambda^{-d}$. For $h \in \mathcal{P}$, let D_h be the distribution of the random variable $x + \eta h(\pi(x))v$ where $x \sim D_0$. Let $\mathcal{D} = \{D_h : h \in \mathcal{P}\}$.

Thus, $\mathcal{D}$ is a family of η-perturbations of D_0. Furthermore, since $f(x+\eta v) - f(x-\eta v) = 2\eta\Lambda^d = 2$ for all $x \in \mathrm{supp}(D_0) = X$, we have that f is a Δ-separator for $\mathcal{D}$ where $\Delta = 2$.

To show the first claim of the theorem, that there is a $(0, \mathrm{NN}_{\Lambda,3})$-compression of f over any $D \in \mathcal{D}$, we first observe that for any $h \in \mathcal{P}$ and $x \in \mathrm{supp}(D_h)$, we have

$$f(x) = \Lambda^d \eta h(\pi(x)) = h(\pi(x))\,.$$

The claim then follows immediately from Lemma 2, for appropriate choice of $n = \Omega(\Lambda)$.

To bound the SQ complexity of the $(f, \mathrm{NN}_{\Lambda,d-1}, \mathcal{D})$-compression problem, we first suppose $g \in \mathrm{NN}_{\Lambda,d-1})$ is a $(1/16, \mathrm{NN}_{\Lambda,d-1})$-compression of f over some $D_h \in \mathcal{D}$. By Lemma 1, given inputs f and g, there is an an efficient algorithm for estimating probabilities $\tilde{p}(x)$ of points under a distribution $\tilde{D}$, which is an η-perturbation of D_0 satisfying $\mathrm{D_{TV}}(\tilde{D}, \tilde{D}) < 1/4$. Since $\tilde{D}$ is an η-perturbation of D_0, its support includes exactly one of $x + \eta v$ and $x - \eta v$ for each $x \in X$. Let $\tilde{h}(x) = 1$ if $\tilde{p}(x + \eta v) > 0$ and $\tilde{h}(x) = -1$ otherwise. Then $\Pr_{x \sim D_0}(\tilde{h}(x) \neq h(x)) < 1/4$. In particular, given such a $g \in \mathrm{NN}_{\Lambda,d-1}$, we can learn h to within accuracy $3/4$ without any additional access to D_h. Hence, by Lemma 3, finding such a g has total complexity $\exp(\Omega(n)) = \exp(\Omega(\Lambda))$.

Proof (Proof of Theorem 2). The proof is essentially the same as for Theorem 1. We replace the parity functions with their noisy versions, with noise rate $\varepsilon/2$. A neural network computing the (non-noisy) parity, as from Lemma 2, will be an ε-approximation of the noisy parity with high probability. The result again follows by Lemma 1.

4 Local vs. Global Compression

The proof of Theorem 3 is a straightforward application of existing depth separation theorems for neural networks. Several versions of depth separations of 2-layer from 3-layer networks are known [7,26,11].

Theorem 4 (Eldan and Shamir [11, Theorem 1]). *There is a constant $c > 0$ such that for every n there is a probability distribution D on $\mathbb{R}^n$ and a map $g : \mathbb{R}^n \to \mathbb{R}$ with a neural network representation $g \in \mathrm{NN}_{\mathrm{poly}(n),3}$, such that if there exists a $(c, \mathrm{NN}_{\Lambda,2})$-compression of g, then $\Lambda = \exp(\Omega(n))$.*

Proof (Proof of Theorem 3). Let D be the distribution and $g : \mathbb{R}^n \to \mathbb{R}$ be the neural network in $\mathrm{NN}_{\Lambda,3}$ given by Theorem 4. We define the network $f : \mathbb{R}^{n \times (d-3)} \to \mathbb{R}$ in $\mathrm{NN}_{TODO,d}$ using g as a gadget. Specifically, for $1 \leq i \leq d-3$, let $f^(i)(x) : \mathbb{R}^n \to \mathbb{R}$ be the network in $\mathrm{NN}_{\Lambda,d}$ be the network whose first i layers each compute the identity map on $\mathbb{R}^n$, whose $i+1$ through $i+3$ layers are identical to those of g, and whose subsequent $d-i-3$ layers each compute the identity map on $\mathbb{R}$. Let f be the neural network given by the product of the $f^{(i)}$ for $1 \leq i \leq d-3$. That is, the jth layer of f has as its input space the Cartesian product of the input spaces of the jth layers of the networks $f^{(i)}$ for $1 \leq i \leq d-3$, and similarly for the output spaces, and map computed on the jth layer is the Cartesian product of the maps of the jth layers of the networks $f^{(i)}$. The input distribution of f is D^{d-3}.

Clearly f has an exact representation in $\mathrm{NN}_{(d-3)\Lambda,3}$ as product of g with itself $d-3$ times, so a $(0, \mathrm{NN}_{(d-3)\Lambda,3})$-compression of f over D^{d-3} exists. We argue that f is not $(O(1/d), 2, 3)$-locally compressible over D^{d-3}. Letting $h = f^{(i)}$ be the ith constituent network of f, we have $h^{[i:i+3]} = g$. Furthermore, the distribution

induced by D on the ith layer of h is simply D. Therefore by Theorem 4, there does not exist a $(c, \mathrm{NN}_{\mathrm{poly}(\Lambda),2})$-compression of of $h^{[i:i+3]}$ over this distribution, for some absolute constant $c \geq 0$, and so the same is true of f. Hence, f is not $(c, 2, 3)$-locally compressible over D^{d-3}.

It is reasonable to conjecture that similar statements larger notions of local compressibility, e.g., $(c, 3, 5)$-local compression rather than $(c, 2, 3)$-local compression. Such results would similarly follow from depth separation theorems deeper neural networks. However, proving such separation theorems is a major open problem.

5 Outlook

assumptions for feasibility: - smooth in neighborhood of distribution
equivalent to robustness to adversarial inputs?
path to algorithmic guarantees - make use of implicit bias / implicit regularization? foreshadow Will's current project

References

1. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Advances in Neural Information Processing Systems (NIPS). pp. 2654–2662 (2014)
2. Blum, A., Frieze, A., Kannan, R., Vempala, S.: A polynomial-time algorithm for learning noisy linear threshold functions. Algorithmica **22**(1-2), 35–52 (1998)
3. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM (JACM) **50**(4), 506–519 (2003)
4. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 535541 (2006)
5. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: Advances in Neural Information Processing Systems. pp. 742–751 (2017)
6. Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Olukotun, K., Ng, A.Y.: Map-reduce for machine learning on multicore. In: Advances in neural information processing systems. pp. 281–288 (2007)
7. Daniely, A.: Depth separation for neural networks. In: Conference on Learning Theory. pp. 690–696 (2017)
8. Denil, M., Shakibi, B., Dinh, L., Ranzato, M., De Freitas, N.: Predicting parameters in deep learning. In: Advances in Neural Information Processing Systems (NIPS). pp. 2148–2156 (2013)
9. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in Neural Information Processing Systems (NIPS). pp. 1269–1277 (2014)
10. Dunagan, J., Vempala, S.: A simple polynomial-time rescaling algorithm for solving linear programs. Mathematical Programming **114**(1), 101–114 (2008)
11. Eldan, R., Shamir, O.: The power of depth for feedforward neural networks. In: Conference on Learning Theory. pp. 907–940 (2016)

12. Feldman, V.: A general characterization of the statistical query complexity. In: Conference on Learning Theory. pp. 785–830 (2017)
13. Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: On agnostic learning of parities, monomials, and halfspaces. SIAM Journal on Computing **39**(2), 606–645 (2009)
14. Feldman, V., Grigorescu, E., Reyzin, L., Vempala, S.S., Xiao, Y.: Statistical algorithms and a lower bound for detecting planted cliques. Journal of the ACM (JACM) **64**(2), 1–37 (2017)
15. Gunasekar, S., Lee, J.D., Soudry, D., Srebro, N.: Implicit bias of gradient descent on linear convolutional networks. In: Advances in Neural Information Processing Systems. pp. 9461–9471 (2018)
16. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems (NIPS). pp. 1135–1143 (2015)
17. Hassibi, B., Stork, D.G., Wolff, G.J.: Optimal brain surgeon and general network pruning. In: Neural Networks. pp. 293–299 (1993)
18. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv:1503.02531 (2015)
19. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv:1405.3866 (2014)
20. Kearns, M.J.: Efficient noise-tolerant learning from statistical queries. In: Proc. 25th ACM Symp. on Theory of Computing (STOC). pp. 392–401 (1993)
21. Kiltz, E., Pietrzak, K., Venturi, D., Cash, D., Jain, A.: Efficient authentication from hard learning problems. Journal of Cryptology **30**(4), 1238–1275 (2017)
22. Klivans, A., Kothari, P.: Embedding hard learning problems into gaussian space. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014) (2014)
23. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems (NIPS). pp. 598–605 (1990)
24. Neyshabur, B., Tomioka, R., Srebro, N.: In search of the real inductive bias: On the role of implicit regularization in deep learning. arXiv preprint arXiv:1412.6614 (2014)
25. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv:1412.6550 (2014)
26. Safran, I., Shamir, O.: Depth-width tradeoffs in approximating natural functions with neural networks. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70, pp. 2979–2987 (2017)
27. Song, L., Vempala, S., Wilmes, J., Xie, B.: On the complexity of learning neural networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 5520–5528 (2017)
28. Soudry, D., Hoffer, E., Nacson, M.S., Gunasekar, S., Srebro, N.: The implicit bias of gradient descent on separable data. The Journal of Machine Learning Research **19**(1), 2822–2878 (2018)
29. Tai, C., Xiao, T., Zhang, Y., Wang, X., Weinan, E.: Convolutional neural networks with low-rank regularization. arXiv:1511.06067 (2015)
30. Vempala, S., Wilmes, J.: Gradient descent for one-hidden-layer neural networks: Polynomial convergence and sq lower bounds. In: Conference on Learning Theory. pp. 3115–3117 (2019)
31. Zhu, M., Gupta, S.: To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv:1710.01878 (2017)