Towards a Theoretical Understanding of Hashing-Based Neural Nets

Yibo Lin
yibolin@utexas.edu
UT-Austin

Zhao Song
zhaos@g.harvard.edu
Harvard & UT-Austin

Lin F. Yang
lin.yang@princeton.edu
Princeton University

Abstract

Parameter reduction has been an important topic in deep learning due to the ever-increasing size of deep neural network models and the need to train and run them on resource limited machines. Despite many efforts in this area, there were no rigorous theoretical guarantees on why existing neural net compression methods should work. In this paper, we provide provable guarantees on some hashing-based parameter reduction methods in neural nets. First, we introduce a neural net compression scheme based on random linear sketching (which is usually implemented efficiently via hashing), and show that the sketched (smaller) network is able to approximate the original network on all input data coming from any smooth and well-conditioned low-dimensional manifold. The sketched network can also be trained directly via back-propagation. Next, we study the previously proposed HashedNets architecture and show that the optimization landscape of one-hidden-layer HashedNets has a local strong convexity property similar to a normal fully connected neural network. We complement our theoretical results with empirical verifications.[1]

1 Introduction

In the past decade, deep neural networks have become the new standards for many machine learning applications, including computer vision Krizhevsky et al. (2012); He et al. (2016), natural language processing Zaremba et al. (2014); Gehring et al. (2017), speech recognition Graves et al. (2013); Amodei et al. (2016), robotics Lillicrap et al. (2015), game playing Silver et al. (2016, 2017), etc. Such model usually contains an enormous number of parameters, which is often much larger than the number of available training samples. Therefore, these networks are usually trained on modern computer clusters which have a huge amount of memory and computation power. On the other hand, there is an increasing need to train and run personalized machine learning models on mobile and embedded devices instead of transferring mobile data to a remote computation center on which all the computations are performed. This is because real-time processing of deep learning models on mobile devices brings the benefits of better privacy and less Internet bandwidth. However, mobile devices like smart phones do not have the memory or computation capability of training large neural networks or even storing these models.

These trends motivate the study of *neural network compression*, with the goal of reducing the memory overhead required to train, store and run neural networks. There is a recent line of research in this direction, for example Chen et al. (2015); Iandola et al. (2016); Han et al. (2016). Despite their empirical effectiveness, there is little theoretical understanding on why these methods perform well.

The goal of this paper is to bridge the gap between theory and practice in neural network compression. Our focus is on *hashing-based methods*, which have been studied empirically in e.g. Chen et al. (2015, 2016), with the hope that the randomness in hash functions helps preserve the properties of neural networks despite a reduction in the number of effective parameters. We make this intuition formal by giving theoretical guarantees on the *approximation power* and the *parameter recovery* of such networks.

First, we propose a neural net compression scheme based on random linear sketching, which can be efficiently implemented using a hash function. Simi-

[1] The full version of this paper is available at http://arxiv.org/pdf/1812.10244.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

lar idea has been proposed in Kasiviswanathan et al. (2017) and demonstrated high performance empirically, but no formal theoretical guarantee was known. We show that such compression has strong approximation power. Namely, the small network obtained after sketching can approximate the original network on *all* input data coming from *any low-dimensional manifold* with some regularity properties. The sketched network is also directly trainable via back-propagation. In fact, sketching is a principled technique for dimensionality reduction, which has been shown to be very powerful in solving various problems arising in statistics Raskutti and Mahoney (2016); Wang et al. (2017) and numerical linear algebra Woodruff (2014). Given its theoretical success, it is natural to ask whether sketching can be applied to the context of neural net compression with theoretical guarantees. Our result makes partial progresses on this question.

Next we study HashedNets, a simple method proposed in Chen et al. (2015) which appears to perform well in practice. HashedNets directly applies a random hash function on the connection weights in a neural net and to enforce all the weights mapped to the same hash bucket to take the same value. In this way the number of trainable parameters is reduced to be the number of different hash buckets, and training can still be performed via back-propagation while taking the weight sharing structure into account. From the perspective of optimization, we show that the training objective for a one-hidden-layer hashed neural net has a local strong convexity property, similar to that of a normal fully connected network Zhong et al. (2017b). Additionally, we can apply the initialization algorithm in Zhong et al. (2017b) to obtain a good initialization for training. Therefore it implies that the parameters in one-hidden-layer HashedNets can be provably recovered under milde assumptions.

Below we describe our contributions in more detail.

Approximation Power Our result on the approximation power of sketched nets is based on a classical concept, "subspace embedding", which originally appears in numerical linear algebra Sarlós (2006). Roughly speaking, it says that there exist a wide family of random matrices $S \in \mathbb{R}^{s\times n}$, such that for *any* d-dimensional subspace $\mathcal{U} \subset \mathbb{R}^n$, with probability $1-\delta$ we have $\langle Sx, Sx'\rangle = \langle x, x'\rangle \pm \epsilon\|x\|_2\|x'\|_2$ for all $x, x' \in \mathcal{U}$, provided $s = \Omega\left((d+\log 1/\delta)/\epsilon^2\right)$. This result means that the inner product between every two points in a subspace can be approximated *simultaneously* after applying a random *sketching matrix* S, which is interesting if $s \ll n$. There has been a line of work trying to do subspace embedding using different sketching matrices (e.g. Nelson and Nguyên (2013); Cohen (2016)). *Sparse matrices* are of particular interests, since for a sparse matrix S, one can compute Sx more efficiently. For example, Nelson and Nguyên (2013) showed that it is possible to construct S with only $\widetilde{O}(1/\epsilon)$ nonzero entries per column, which significantly improves the trivial upper bound $\widetilde{O}(d/\epsilon^2)$. Furthermore, many of these sketching matrices can be efficiently implemented by k-wise independent hash functions where k is very small, which only takes a small amount of space to store, and multiplying S with a vector can be computed efficiently.

We extend the idea of subspace embedding to deep learning and show that a feed-forward fully connected network with Lipschitz-continuous activation functions can be approximated using random sketching on all input data coming from a low-dimensional subspace. Below we describe our result for one-hidden-layer neural nets, and this can be generalized to multiple layers.

Consider a one-hidden-layer neural net with input dimension n and k hidden nodes. It can be parameterized by a weight matrix $W \in \mathbb{R}^{n\times k}$ and a weight vector $v \in \mathbb{R}^k$, and the function this network computes is $x \mapsto v^\top \phi(W^\top x)$, where $x \in \mathbb{R}^n$ is the input, and ϕ should be viewed as a nonlinear activation function acting coordinate-wise on a vector. Our result says that under appropriate assumptions, one can choose a random sketching matrix $S \in \mathbb{R}^{s\times n}$, such that for any d-dimensional subspace $\mathcal{U} \subset \mathbb{R}^n$, we have

$$\left|v^\top \phi(W^\top x) - v^\top \phi(W^\top S^\top S x)\right| \le \epsilon, \forall x \in \mathcal{U}, \|x\|_2 \le 1.$$

This result essentially says that the weight matrix $W^\top$ can be replaced by $W^\top S^\top S$, which has rank s. When $s < n$, this means that the effective number of parameters can be reduced from kn to ks. As we mentioned, the sketching matrix S can be implemented by hash functions in small space and multiplying it with a vector is efficient. The sketched network is also directly trainable, because we can train the $s \times k$ matrix $\widehat{W} = SW$, regarding another factor S in the decomposition $W^\top S^\top S = \widehat{W} S$ as a known layer.

This result can be generalized to multi-layer neural nets, and we present the details in Section 3. We also note that our result can be easily generalized to low-dimensional manifolds under some regularity condition (see Definition 2.3 in Baraniuk and Wakin (2009)), which is a much more realistic assumption on data.

Parameter Recovery. It is known that training a neural net is NP-hard in the worst case, even if it only has 3 hidden nodes Blum and Rivest (1993). Recently, there has been some theoretical progress on understanding the optimization landscapes of shallow neural nets under special input distributions. In particu-

lar, Zhong et al. (2017b) gave a recovery guarantee for one-hidden-layer neural nets. They showed that if the input distribution is Gaussian and the ground-truth weight vectors corresponding to hidden nodes are linearly independent, then the true parameters can be recovered in polynomial time given finite samples. This was proved by showing that the training objective is locally strongly convex and smooth around the ground-truth point, together with an initialization method that can output a point inside the locally "nice" region. In this work, we show that local strong convexity and smoothness continue to hold if we replace the fully connected network by HashedNets which has a weight sharing structure enforced by a hash function. We present this result in Section 4.

1.1 Related Works

Parameter Reduction in Deep Learning. There has been a series of empirical works on reducing the number of free parameters in deep neural networks: Denil et al. (2013) show a method to learn low-rank decompositions of weight matrices in each layer, Chen et al. (2015) propose an approach to use a hash function to enforce parameter sharing, Cheng et al. (2015) adopt a circulant matrix structure for parameter reduction, Sindhwani et al. (2015) study a more general class of structured matrices for parameter reduction.

Sketching and Neural Networks. Daniely et al. (2016) show that any linear or sparse polynomial function on sparse binary data can be computed by a small single-layer neural net on a linear sketch of the data. Kasiviswanathan et al. (2017) apply a random sketching on weight matrices/tensors, but they only prove that given a fixed layer input, the output of this layer using sketching matrices is an unbiased estimator of the original output of this layer and has bounded variance; however, this does not provide guarantees on the approximation power of the whole sketching-based deep net.

Subspace Embedding. Subspace embedding Sarlós (2006) is a fundamental tool for solving numerical linear algebra problems, e.g. linear regression, matrix low-rank approximation Clarkson and Woodruff (2013); Nelson and Nguyên (2013); Razenshteyn et al. (2016); Song et al. (2017b), tensor low-rank approximation Song et al. (2019). See also Woodruff (2014) for a survey on this topic.

Recovery Guarantee of Neural Networks. Since learning a neural net is NP-hard in the worst case Blum and Rivest (1993), many attempts have been made to design algorithms that learns a neural net provably in polynomial time and sample complexity under additional assumptions, e.g., Sedghi and Anandkumar (2014); Zhang et al. (2015); Janzamin et al. (2015); Goel et al. (2017); Goel and Klivans (2017a,b). Another line of work focused on analyzing (stochastic) gradient descent on shallow networks for Gaussian input distributions, e.g., Brutzkus and Globerson (2017); Zhong et al. (2017a,b); Tian (2017); Li and Yuan (2017); Du et al. (2017); Soltanolkotabi (2017).

Other Related Works Instead of understanding the parameter reduction as our work, there are several results working on developing over-parameterization theory of deep ReLU neural networks, e.g. Allen-Zhu et al. (2018a,b). Thirty years ago, Blum and Rivest proved training neural network is NP-hard Blum and Rivest (1993). Later, neural networks have been shown hard in several different perspectives Klivans and Sherstov (2009); Livni et al. (2014); Daniely (2016); Daniely and Shalev-Shwartz (2016); Goel et al. (2017); Song et al. (2017a); Katz et al. (2017); Weng et al. (2018); Manurangsi and Reichman (2018) in the worst case regime.

Arora et al. proved a stronger generalization for deep nets via a compression approach Arora et al. (2018). There is a long line of works targeting on explaining GAN from theoretical perspective Arora and Zhang (2017); Arora et al. (2017b,a); Bora et al. (2017); Li et al. (2018); Santurkar et al. (2018); Van Veen et al. (2018); Xiao et al. (2018). There is also a long line of provable results about adversarial examples Madry et al. (2017); Bubeck et al. (2018b,a); Weng et al. (2018); Schmidt et al. (2018); Tran et al. (2018).

2 Preliminaries

For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. Let $a \pm b$ represent any number in the interval $[a-b, a+b]$. For any vector $x \in \mathbb{R}^n$, we use $\|x\|_2$, $\|x\|_1$ and $\|x\|_\infty$ to denote its ℓ_2, ℓ_1 and ℓ_∞ norms, respectively. For $x, y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the standard Euclidean inner product $x^\top y$.

For a matrix A, let $\det(A)$ denote its determinant (if A is a square matrix), let $A^\dagger$ denote the Moore-Penrose pseudoinverse of A, and let $\|A\|_F$ and $\|A\|_2$ denote respectively the Frobenius norm and the spectral norm of A. Denote by $\sigma_i(A)$ the i-th largest singular value of A. We use $\mathrm{nnz}(A)$ to denote the number of non-zero entries in A.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C.

We define the ℓ_1 and ℓ_2 balls in $\mathbb{R}^n$ as: $\mathcal{B}_1(B,n) = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq B\}, \mathcal{B}_2(B,n) = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq B\}$. We also need the definitions of Lipschitz-continuous functions and k-wise independent hash families.

Definition 2.1. *A function $f : \mathbb{R} \to \mathbb{R}$ is L-Lipshitz continuous, if for all $x_1, x_2 \in \mathbb{R}$, $|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$.*

Definition 2.2. *A family of hash functions $\mathcal{H} \subseteq \{h \mid h : U \to [B]\}$ is said to be k-wise independent if for any $x_1, \ldots, x_k \in U$ and any $y_1, \ldots, y_k \in [B]$ we have $\Pr_{h\sim\mathcal{H}}[h(x_i) = y_i, \forall i \in [k]] = \frac{1}{B^k}$.*

3 The Approximation Power of Parameter-Reduced Neural Networks

In this section, we study the approximation power of parameter-reduced neural nets based on hashing. Any weight matrix W in a neural net acts on a vector x as Wx. We replace Wx by $WS^\top Sx$ for some sketching (#rows < #columns) matrix S defined in the following section. Then the new weight matrix $WS^\top$ has much fewer parameters. We show that if S is chosen properly as a *subspace embedding* (formally defined later in this section), the sketched network can approximate the original network on all inputs coming from a low-dimensional subspace or manifold. Our sketching matrix is chosen as a Johnson-Lindenstrauss (JL) Johnson and Lindenstrauss (1984) transformation matrix. In Section 3.1, we provide some preliminaries on subspace embedding. In Sections 3.2 and 3.3, we present our result on one-hidden-layer neural nets. Then in Section 3.4 we extend this result to multi-layer neural nets and show a similar approximation guarantee. This provides a theoretical guarantee for hashing-based parameter-reduced networks used in practice.

3.1 Subspace Embedding

We first present some basic definitions of sketching and subspace embedding. These mathematical tools are building blocks for us to understand parameter-reduced neural networks.

Definition 3.1 (Subspace Embedding)**.** *A $(1 \pm \epsilon)$ ℓ_2-subspace embedding for the column space of an $n \times d$ matrix U is a matrix S for which for all $x \in$ colspan(U), $\|Sx\|_2^2 = (1 \pm \epsilon)\|x\|_2^2$,or equivalently, for all $x, x' \in$ colspan(U), $\langle Sx, Sx'\rangle = \langle x, x'\rangle \pm \epsilon\|x\|_2\|x'\|_2$.*

Constructions of subspace embedding can be found in e.g. Nelson and Nguyên (2013) where the following theorem is presented.

Theorem 3.2 (Nelson and Nguyên (2013))**.** *There is a $(1 \pm \epsilon)$ oblivious[2] ℓ_2-subspace embedding for $n \times d$ matrix U with $s = d \cdot \text{poly}\log(d/(\epsilon\delta))/\epsilon^2$ rows and error probability δ. Further, $S \cdot U$ can be computed in time $O(\text{nnz}(U)\,\text{poly}\log(d/(\epsilon\delta))/\epsilon)$. We call S a* SparseEmbedding *matrix.*

There are also other subspace embedding matrices, e.g., CountSketch. We provide additional definitions and examples in Section B.1.

Remark 3.3. *We remark that the subspace embedding in Definition 3.1 naturally extends to low dimensional manifolds. For example, for a d-dimensional Riemannian submanifold of $\mathbb{R}^n$ with volumn V and geodesic covering regularity R (see Definition 2.3 in Baraniuk and Wakin (2009)), Theorem 3.2 holds by replacing d with $Rd\log(V)$. For ease of presentation, we only present our results for subspaces. All our results can be extended to low-dimensional manifolds satisfying regularity conditions.*

3.2 One Hidden Layer - Part I

We consider one-hidden-layer neural nets in the form $\sum_i v_i\phi_i(w_i^\top x)$, where x is the input vector, w_i is a weight vector, v_i is a weight scalar, and $\phi_i : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function. In this subsection, we show how to sketch the weights between the input layer and the hidden layer with guaranteed approximation power. The main result is Theorem 3.4 and its proofs are in the full version Lin et al. (2018).

Theorem 3.4. *Given parameters $n_2 \geq 1, n_1 \geq 1, \epsilon \in (0,1), \delta \in (0,1)$ and $n = \max(n_2, n_1)$. Given n_2 activation functions $\phi_i : \mathbb{R} \to \mathbb{R}$ that are L-Lipshitz-continuous, a fixed matrix $U \in \mathbb{R}^{n_1\times d}$, weight matrix $W \in \mathbb{R}^{n_1\times n_2}$ with $\|W\|_2 \leq B$, $v \in \mathbb{R}^{n_2}$ with $\|v\|_1 \leq B$. Choose a* SparseEmbedding *matrix $S \in \mathbb{R}^{s\times n_1}$ with $s = O(dL^2B^4A^2\epsilon^{-2}\,\text{poly}\log(nLBA/(\epsilon\delta)))$, then with probability $1-\delta$, we have : for all $x \in$ colspan$(U) \cap \mathcal{B}_2(A, n_1)$,*

$$|\langle v, \phi(W^\top x)\rangle - \langle v, \phi(W^\top S^\top Sx)\rangle| \leq \epsilon.$$

3.3 One-hidden layer - Part II

In this section, we show the approximation power of the compressed network if the weight matrices of both the input layer and output layer are sketched. One of the core idea in the proof is a recursive ϵ-net argument, which plays a crucial role in extending the result to multiple hidden layer. The goal of this section is to prove the following theorem and present the recursive ϵ-net argument.

[2]The construction of S is oblivious to the subspace U.

Theorem 3.5. *Given parameters $n_2 \geq 1, n_2 \geq 1, \epsilon \in (0,1), \delta \in (0,1)$ and $n = \max(n_2, n_1)$. Given n_2 activation functions $\phi_i : \mathbb{R} \to \mathbb{R}$ with L-Lipshitz and normalized by $1/\sqrt{n_2}$, a fixed matrix $U \in \mathbb{R}^{n_1 \times d}$, and weight matrix $W \in \mathbb{R}^{n_2 \times n_1}$ with $\|W\|_2 \leq B$, $v \in \mathbb{R}^{n_2}$ with $\|v\|_2 \leq B$. Choose a* SparseEmbedding *matrix $S_1 \in \mathbb{R}^{s_1 \times n_1}$ and $S_2 \in \mathbb{R}^{s_2 \times n_2}$ with $s_1, s_2 = O(dL^2B^4A^2\epsilon^{-2} \operatorname{poly}\log(nLBA/(\epsilon\delta)))$, then with probability $1-\delta$, we have : for all $x \in \operatorname{colspan}(U) \cap \mathcal{B}_2(A, n_1)$,*

$$\left|\langle v, \phi(W^\top x)\rangle - \langle S_2 v, S_2 \phi(W^\top S_1^\top S_1 x)\rangle\right| \leq \epsilon.$$

The high level idea is as follows. Firstly, we prove that for any fixed input x, the theorem statement holds with high probability. Then we build an sufficiently fine ϵ-net over the input space of x and argue that our statement holds for every input point x from the ϵ-net. Condition on this event, the statement holds by applying the Lipshitz continuity of the activation function. The detailed proof is presented in the full version Lin et al. (2018).

3.4 Multiple hidden layer

In this section, we generalize our approximation power result to a multi-layer neural network and delay the proofs to the Appendix of the full version Lin et al. (2018). Inspired by the batch normalization Ioffe and Szegedy (2015), which has been widely used in practice[3], we make an additional assumption by requiring the activations to be normalized by $1/\sqrt{n_{j+1}}$ at each layer j. The way we deal with multiple hidden layers is, first recursively argue an ϵ-net can be constructed for all the layers with the same size. Then we use triangle inequality to split error into $q+1$ terms and bounding them separately. The result is the following theorem.

Theorem 3.6. *Given parameters $q \geq 1$, $n_j \geq 1, \forall j \in [q+1], \epsilon \in (0,1), \delta \in (0,1)$ and $n = \max_{j \in [q+1]} n_j$. For each $j \in [q]$, for each $i \in [n_{j+1}]$ let $\phi_{j,i} : \mathbb{R} \to \mathbb{R}$ denote an activation function with L-Lipshitz and normalized by $1/\sqrt{n_{j+1}}$. Given a fixed matrix $U \in \mathbb{R}^{n_1 \times d}$, q weight matrices $W_j \in \mathbb{R}^{n_{j+1} \times n_j}, \forall j \in [q]$ with (the i_j-th column of W_j) $w_{j,i_j} \in \mathcal{B}_2(B, n_j)$, a weight vector $v \in \mathbb{R}^{n_{j+1}}$ with $v \in \mathcal{B}_2(B, n_{j+1})$. For each $j \in [q+1]$, we choose a* SparseEmbedding *matrix $S_j \in \mathbb{R}^{s_j \times n_j}$ with*

$$s_j = O(dq^2L^{2q}B^{2q+2}A^2\epsilon^{-2} \operatorname{poly}\log(nqLBA/(\epsilon\delta))).$$

Then with probability $1-\delta$, we have : for all $x \in \operatorname{colspan}(U) \cap \mathcal{B}_2(A, n_1)$,

$$|\langle v, f^{(q)}(x)\rangle - \langle S_{q+1} v, S_{q+1} \widetilde{f}^{(q)}(x)\rangle| \leq \epsilon,$$

where $f^{(q)}(x)$ and $\widetilde{f}^{(q)}(x)$ are defined recursively. The base case is $f^{(0)}(x) = \widetilde{f}^{(0)}(x) = x$, and for $j > 0$,

$$f^{(j)}(x) = \phi_j(W_j^\top f^{(j-1)}(x)), \forall j \in [q]$$
$$\text{and } \widetilde{f}^{(j)}(x) = \phi_j(W_j^\top S_j^\top S_j \widetilde{f}^{(j-1)}(x)), \forall j \in [q].$$

Note that similar results also hold for the case without using S_{q+1}. In other words, we only choose q matrices for q hidden layers.

4 Recovery Guarantee

In this section, we study the recovery guarantee of parameter-reduced neural nets. In particular, we study whether (stochastic) gradient descent can learn the true parameters in a one-hidden-layer HashedNets when starting from a sufficiently good initialization point, under appropriate assumptions. We show that even under the special weight sharing structure depicted by the hash function, the resulting neural net still has sufficiently nice properties - namely, local strong convexity and smoothness around the minimizer. Our proof technique is by reducing our case to that of the fully connected network studied in Zhong et al. (2017b). After that, the recovery guarantee follows similarly. We present our result here and give the detailed proof in the full version Lin et al. (2018).

We consider the following regression problem : given a set of m samples $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\} \subset \mathbb{R}^n \times \mathbb{R}$. Let $\mathcal{D}$ be an underlying distribution over $\mathbb{R}^n \times \mathbb{R}$ with parameters $w^* \in \mathbb{R}^B$ and $v^* \in \mathbb{R}^k$, such that each sample $(x, y) \in S$ is sampled i.i.d. from this distribution, with $x \sim \mathcal{N}(0, I), y = \sum_{i=1}^k v_i^* \cdot \phi(\sum_{j=1}^n w^*_{h(i,j)} \cdot x_j)$. Here $h : [k] \times [n] \to [B]$ is a random hash function drawn from a t-wise independent hash family $\mathcal{H}$, where $t = \Theta(\log(nk))$, and ϕ is an activation function.

Note that w^* has a corresponding matrix $\widehat{W}^* \in \mathbb{R}^{k \times n}$ defined as $\widehat{W}^*_{ij} = w^*_{h(i,j)}$, which is the actual weight matrix in the HashedNets with a weight sharing structure.

Our goal is to recover the ground-truth parameters w^*, v^* given the sample S. Note that how to recover v^* has been discussed in Zhong et al. (2017a,b); their method also applies to our situation. Therefore we focus on recovering w^* in this section, assuming v^* is known.

For a given weight vector $w \in \mathbb{R}^B$, we define its *expected risk* and *empirical risk* as

$$F_{\mathcal{D}}(w) = \frac{1}{2} \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{D}} \left[\left(\sum_{i=1}^k v_i^* \cdot \phi \left(\sum_{j=1}^n w_{h(i,j)} \cdot x_j \right) - y \right)^2 \right],$$

[3] https://www.tensorflow.org/api_docs/python/tf/nn/batch_normalization

$$F_S(w) = \frac{1}{2} \sum_{(x,y)\sim S} \left[\left(\sum_{i=1}^{k} v_i^* \cdot \phi \left(\sum_{j=1}^{n} w_{h(i,j)} \cdot x_j \right) - y \right)^2 \right].$$

We first show a structural result for t-wise independent hash family, which says the pre-image of each bucket is pretty balanced.

Lemma 4.1 (Concentration of hashing buckets, part of Lemma C.12)**.** *Given integers N and $B \lesssim N/\log N$. Let $h : [N] \to [B]$ denote a t-wise independent hash function such that $\Pr_{h\sim\mathcal{H}}[h(i) = j] = 1/B, \forall i \in [N], \forall j \in [B]$. Then, if $t = \Theta(\log N)$, with probability at least $1 - 1/\operatorname{poly}(N)$, we have for all $j \in [B]$, $0.9N/B \le \sum_{i=1}^{N} \mathbf{1}_{h(i)=j} \le 1.1N/B$.*

The previous work Zhong et al. (2017b) showed that a fully connected network whose ground-truth weight matrix $W^* \in \mathbb{R}^{k\times n}$ has rank k has local strong convexity and smoothness around W^* in its loss function (see their Lemma D.3).

Using Lemma 4.1 as our core reduction tool, we can reduce HashedNets to a fully connected net and obtain the following result:

Theorem 4.2 (Local strong convexity and smoothness)**.** *Suppose* $\operatorname{rank}(\widehat{W}^*) = k$*. Then we have*

$$0.5(kn/B) \cdot A_{\min} \cdot I \preceq \nabla^2 F_{\mathcal{D}}(w^*) \preceq 2(kn/B) \cdot A_{\max} \cdot I,$$

where $A_{\max}$ and $A_{\min}$ are positive parameters that depend on $\widehat{W}^*$ and the activation function ϕ.

Remark 4.3. *A crucial assumption in Theorem 4.2 is that the weight matrix $\widehat{W}^*$ has rank k. In Section 5, we use numerical experiment to verify this assumption in learned HashedNets.*

For the empirical risk F_S, we can show that its Hessian $\nabla^2 F_S(w^*)$ at the optimal point also satisfies similar properties given enough samples. See Theorem C.7 for details.

Using the tensor initialization method in Zhong et al. (2017b), we can find a point in the locally "nice region" around w^*, and then we can show that gradient descent on the empirical risk function F_S converges linearly to w^*. The result is summarized as follows.

Theorem 4.4 (Recovery guarantee)**.** *There exist parameters γ_1 and γ_2 that depend on $\widehat{W}^*$ and ϕ such that the following is true. Let w^c be any point satisfying $\|w^c - w^*\|_2 \le \gamma_1\|w^*\|_2$, and let S denote a set of i.i.d. samples from the distribution $\mathcal{D}$. Define $m_0 = \Theta(\frac{kn}{B} A_{\min})$ and $M_0 = \Theta(\frac{kn}{B} A_{\max})$ where $A_{\max}$ and $A_{\min}$ are the same ones in Theorem 4.2. For any $t \ge 1$, if we choose $|S| \ge n \cdot \operatorname{poly}(\log n, t) \cdot k^2\gamma_2$ and perform gradient descent with step size $1/M_0$ on F_S and obtain the next iterate, $\widetilde{w} = w^c - \frac{1}{M_0}\nabla F_S(w^c)$, then with probability at least $1 - n^{-\Omega(t)}$, we have $\|\widetilde{w} - w^*\|_2^2 \le (1 - m_0/M_0)\|w^c - w^*\|_2^2$.*

The above theorem states that once a constantly-accurate initialization point is specified, we can obtain a solution up to precision $\exp(-\operatorname{poly}(n))$ in a polynomial number of gradient descent iterations. This concludes the recovery guarantee. We give the formal statements and proofs in Section C.

5 Experiments

In this section, we perform some simple experiments on MNIST dataset to evaluate the performance of HashedNets, as well as empirically verify the full rank assumption (as in Theorem 4.2) on weight matrices in HashedNets. Each image in MNIST dataset has a dimensionality of 28×28. The HashedNets in the experiment have single-hidden-layer, i.e., two fully connected layers. To validate the effectiveness of HashedNets, we construct two baselines.

- **SmallNets**. A single-hidden-layer network is constructed with the same amount of effective weights as that of HashedNets. For example, for a HashedNets with 1000 hidden units in the hidden layer with compression ratio 64, a corresponding SmallNets have $\lceil \frac{1000}{64} \rceil = 16$ hidden units in the hidden layer.
- **ThinNets**. A two-hidden-layer network is constructed with the same amount of effective weights as that of HashedNets. By replacing the first fully connected layer in HashedNets with a thin hidden layer, a same amount of weights can be achieved. For example, for a HashedNets with 1000 hidden units in the hidden layer with compression ratio 64, a corresponding ThinNets have $\lceil \frac{784\times1000}{(784+1000)\times64} \rceil = 7$ hidden units for the first hidden layer and 1000 hidden units for the second hidden layer.

The accuracy of HashedNets, SmallNets, and ThinNets is compared under various compression ratios.

The HashedNets were implemented in Torch7 Collobert et al. (2011); Chen et al. (2015) and validated on NVIDIA GTX 1080 GPU. We used 32 bit precision floating point numbers throughout the experiments. Stochastic gradient descent was adopted as the numerical optimizer with a dropout keep rate of 0.9, momentum of 0.9, and a batch size of 50. ReLU was used as the activation function. We ran 1000 epochs for each experiment and experiments on two single-hidden-layer HashedNets with 500 and 1000 hidden units are conducted, respectively. The amount of units

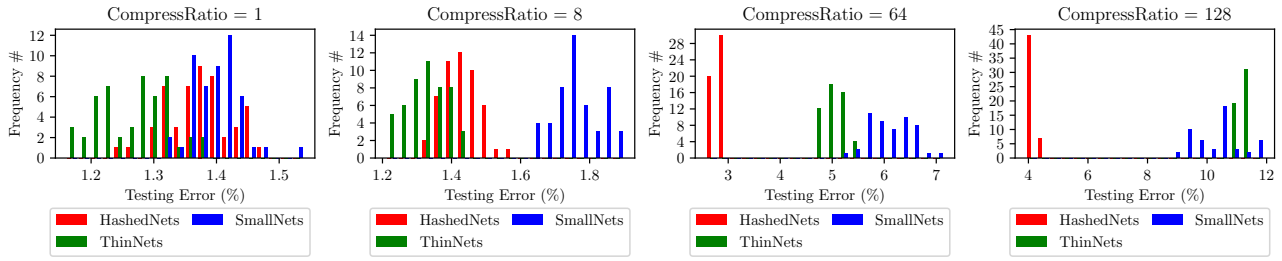

Figure 1: (a) Ratio= 1; (b) ratio= 8; (c) ratio= 64; (d) ratio= 128. We run two one-hidden layer algorithms on MNIST dataset. Comparison of accuracy distribution with different random seeds for HashedNets and SmallNets. Choose 50 random seeds in total. HashedNets have 1000 hidden units in this case.

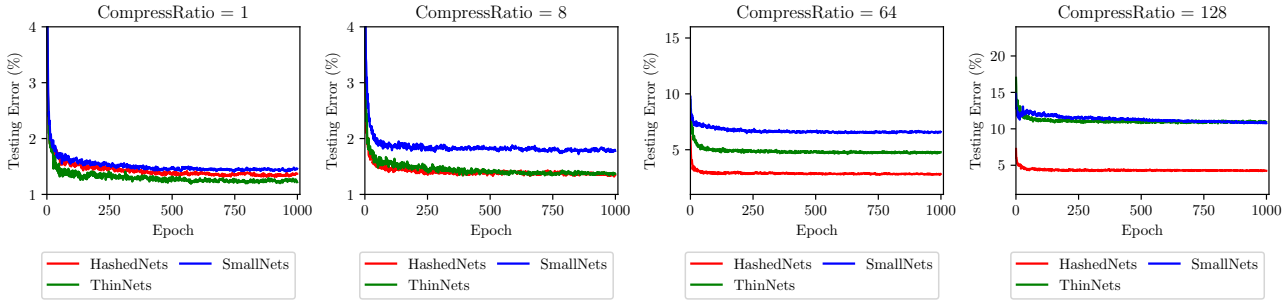

Figure 2: (a) Ratio= 1; (b) ratio= 8; (c) ratio= 64; (d) ratio= 128. The testing error during training of different networks with random seed 100.

in SmallNets and ThinNets is adjusted to match the amount of weights in HashedNets.

For different compression ratios, we plot the distribution of testing errors for 50 runs of HashedNets, ThinNets, and SmallNets with 50 different random seeds, as shown in Figure 1. Due to random initialization, SmallNets still gives different results with independent runs. In Figure 1(a), when the compression ratio is 1, which indicating no compression, the distributions for both HashedNets and SmallNets are very close, i.e., with means of 1.37% and 1.40%, standard deviations of 0.050% and 0.038%, respectively. ThinNets provides slightly better testing error with a mean of 1.27% and a standard deviation of 0.057%. In Figure 1(b), when the compression ratio is 8, HashedNets provides smaller testing errors than that of SmallNets, i.e., with means of 1.43% v.s. 1.76%, and standard deviations of 0.052% v.s. 0.070%, respectively. ThinNets provides slightly better testing errors than that of HashedNets, i.e., with means of 1.32% v.s. 1.44%, and standard deviations of 0.060% v.s. 0.056%. In other words, both HashedNets and ThinNets can achieve higher and more robust accuracy with improvements in both mean and standard deviation than SmallNets for this compression ratio. With the compression ratio increasing to 64, as shown in Figure 1(c), the benefit of HashedNets is more significant. The mean of testing errors for HashedNets degrades to 2.80%, while that for SmallNets increases to 6.09%. The errors for SmallNets are more instable due to larger standard deviations. Meanwhile, HashedNets can also provide better accuracy than ThinNets, i.e., with means of 2.80% v.s. 5.03%, and standard deviations of 0.090% v.s. 0.196%. When the compression ratio is 128, as shown in Figure 1(d), HashedNets achieves a mean accuracy of 4.20% and a standard deviation of 0.116%, which is much better than that of ThinNets, a mean accuracy of 11.09% and a standard deviation of 0.160%, and that of SmallNets, a mean accuracy of 10.28% and a standard deviation of 0.810%. In summary, from the aspect of accuracy degradation, when the compression ratio increases from 1 to 128, there is on average 2.83% degradation in accuracy for HashedNets, while the accuracy of SmallNets degrades by 8.88% and that of ThinNets degrades by 9.82%. ThinNets may achieve comparable error to HashedNet for small compression ratios (e.g., 1 and 8), while for large compression ratio, HashedNet tends to be more stable.

Figure 2 plots the training curves of different networks under different compression ratios with random seed 100. The testing errors align with the observation from Figure 1. That is, HashedNets provides high and stable accuracy across various compression ratios; ThinNets achieves good accuracy for small compression ratios (e.g., 1 and 8, the accuracy is close to that of HashedNets), while degrades significantly with the increase of compression ratios; SmallNets are also very sensitive to large compression ratios like ThinNets.

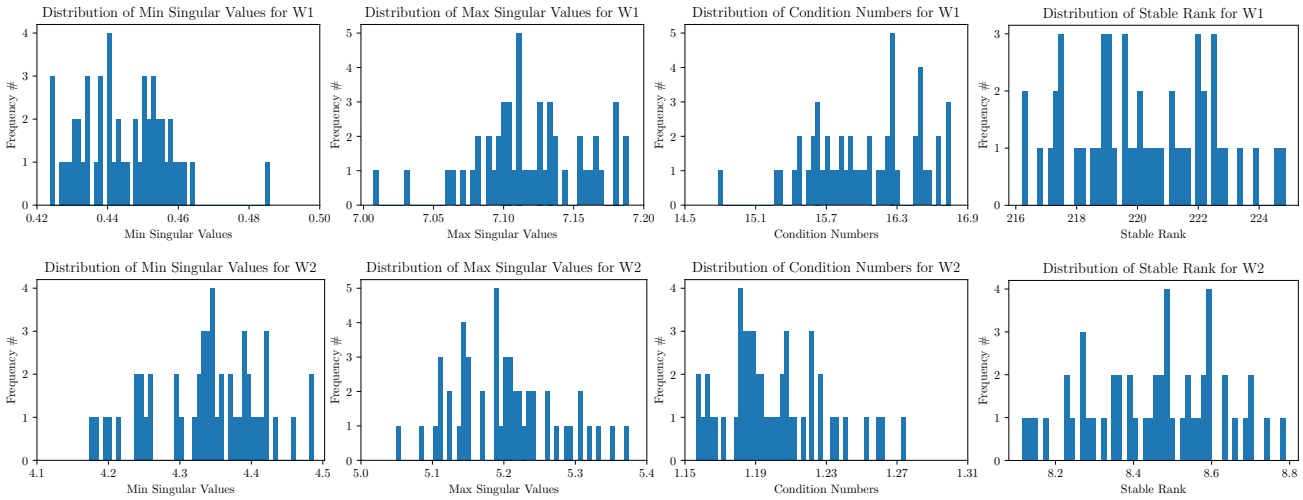

Figure 3: Input dimension is 784. Distributions of singular values, condition numbers, and stable ranks for two weight matrices W_1 and W_2 in HashedNets with 1000 hidden units for 50 random seeds.

We further verify the full rank assumption of weight matrices in HashedNets. Figure 3 plots the distributions of minimum and maximum singular values, condition numbers, and stable ranks of the two weight matrices W_1 and W_2 in HashedNets with 1000 hidden units. The dimensions of W_1 is 1000×784 and that of W_2 is 10×1000. The distributions are extracted from the aforementioned 50 runs. Figure 4 (in the full version Lin et al. (2018)) gives one example of all singular values sorted from large to small in one experiment. All the singular values and condition numbers are distributed in reasonable scales, i.e., neither too close to zero, nor too large. This experiment indicates that the assumption of full rank holds in practice. Same set of figures are also provided for HashedNets with 500 hidden units, as shown in Figure 5 (in the full version Lin et al. (2018)), where the dimensions of W_1 is 500×784 and that of W_2 is 10×500.

6 Conclusion

In this paper, we study the theoretical properties of hashing-based neural networks. We show that (i) parameter-reduced neural nets have uniform approximation power on inputs from any low-dimensional subspace or smooth and well-conditioned manifold; (ii) one-hidden-layer HashedNets have similar recovery guarantee to that of fully connected neural nets. We also empirically explore an alternative compression scheme, ThinNets, which is a very interesting direction for further study, so we plan to explore its property and theoretical insights in the future.

Acknowledgments

This project was done while Zhao was visiting Harvard and hosted by Prof. Jelani Nelson. In 2017, Prof. Jelani Nelson and Prof. Piotr Indyk co-taught a class "Sketching Algorithms for Big Data". One of the class project is, "Understanding hashing trick of neural network." That is the initialization of this project. The authors would like to thank Zhixian Lei for inspiring us of this project.

The authors would like to appreciate Wei Hu for his generous help and contribution to this project.

The authors would like to thank Rasmus Kyng, Eric Price, Zhengyu Wang and Peilin Zhong for useful discussions.

References

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *arXiv preprint*. https://arxiv.org/pdf/1811.03962, 2018a.

Z. Allen-Zhu, Y. Li, and Z. Song. On the convergence rate of training recurrent neural networks. In *arXiv preprint*. https://arxiv.org/pdf/1810.12065, 2018b.

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

S. Arora and Y. Zhang. Do gans actually learn the

distribution? an empirical study. *arXiv preprint arXiv:1706.08224*, 2017.

S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *ICML*. arXiv preprint arXiv:1703.00573, 2017a.

S. Arora, A. Risteski, and Y. Zhang. Theoretical limitations of encoder-decoder gan architectures. *arXiv preprint arXiv:1711.02651*, 2017b.

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In *ICML*. arXiv preprint arXiv:1802.05296, 2018.

R. G. Baraniuk and M. B. Wakin. Random projections of smooth manifolds. *Foundations of computational mathematics*, 9(1):51–77, 2009.

M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 276–287. IEEE, 1994.

A. L. Blum and R. L. Rivest. Training a 3-node neural network is np-complete. In *Machine learning: From theory to applications*, pages 9–28. Springer, 1993.

A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed sensing using generative models. In *ICML*. arXiv preprint arXiv:1703.03208, 2017.

J. Bourgain, V. H. Vu, and P. M. Wood. On the singularity probability of discrete random matrices. *Journal of Functional Analysis*, 258(2):559–603, 2010.

A. Brutzkus and A. Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *ICML*, pages 605–614, 2017.

S. Bubeck, Y. T. Lee, E. Price, and I. Razenshteyn. Adversarial examples from cryptographic pseudo-random generators. *arXiv preprint arXiv:1811.06418*, 2018a.

S. Bubeck, E. Price, and I. Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018b.

M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming*, pages 693–703. Springer, 2002.

W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks in the frequency domain. In *KDD*, pages 1475–1484. ACM, 2016.

Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2857–2865, 2015.

K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 81–90. https://arxiv.org/pdf/1207.6365, 2013.

M. B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Arlington, VA, USA, January 10-12, 2016*, pages 278–287, 2016.

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376 in ., 2011.

A. Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 105–117. ACM, 2016.

A. Daniely and S. Shalev-Shwartz. Complexity theoretic limitations on learning DNFs. In *Conference on Learning Theory (COLT)*, pages 815–830, 2016.

A. Daniely, N. Lazic, Y. Singer, and K. Talwar. Sketching and neural networks. *arXiv preprint arXiv:1604.05753*, 2016.

M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156, 2013.

S. S. Du, J. D. Lee, Y. Tian, B. Poczos, and A. Singh. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779*, 2017.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional Sequence to Sequence Learning. In *ArXiv preprint:1705.03122*, 2017.

S. Goel and A. Klivans. Eigenvalue decay implies polynomial-time learnability for neural networks. In *NIPS*, pages 2189–2199, 2017a.

S. Goel and A. Klivans. Learning depth-three neural networks in polynomial time. *arXiv preprint arXiv:1709.06010*, 2017b.

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

A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1510.00149, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

M. Janzamin, H. Sedghi, and A. Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. In *arXiv preprint*. https://arxiv.org/pdf/1506.08473, 2015.

W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

S. P. Kasiviswanathan, N. Narodytska, and H. Jin. Deep neural network approximation using tensor sketching. *arXiv preprint arXiv:1710.07850*, 2017.

G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, pages 97–117. Springer, 2017.

A. R. Klivans and A. A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

J. Li, A. Madry, J. Peebles, and L. Schmidt. On the limitations of first order approximation in gan dynamics. In *ICML*, 2018.

Y. Li and Y. Yuan. Convergence analysis of two-layer neural networks with relu activation. In *NIPS*, pages 597–607, 2017.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Y. Lin, Z. Song, and L. F. Yang. Towards a theoretical understanding of hashing-based neural nets. *CoRR*, abs/1812.10244, 2018. URL http://arxiv.org/abs/1812.10244.

R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 855–863, 2014.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*. arXiv preprint arXiv:1706.06083, 2017.

P. Manurangsi and D. Reichman. The computational complexity of training ReLU(s). *arXiv preprint arXiv:1810.04207*, 2018.

J. Nelson and H. L. Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 117–126. IEEE, https://arxiv.org/pdf/1211.1002, 2013.

G. Raskutti and M. Mahoney. A statistical perspective on randomized sketching for ordinary least-squares. *JMLR*, 2016.

I. Razenshteyn, Z. Song, and D. P. Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th Annual Symposium on the Theory of Computing (STOC)*, 2016.

D. Salmond, A. Grant, I. Grivell, and T. Chan. On the rank of random matrices over finite fields. *arXiv preprint arXiv:1404.3250*, 2014.

S. Santurkar, L. Schmidt, and A. Madry. A classification-based study of covariate shift in gan distributions. In *International Conference on Machine Learning (ICML)*, pages 4487–4496, 2018.

T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS) , 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 143–152, 2006.

L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Mądry. Adversarially robust generalization requires more data. In *NeurIPS*. arXiv preprint arXiv:1804.11285, 2018.

H. Sedghi and A. Anandkumar. Provable methods for training neural networks with sparse connectivity. *arXiv preprint arXiv:1412.2693*, 2014.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3088–3096, 2015.

M. Soltanolkotabi. Learning relus via gradient descent. In *NIPS*, pages 2004–2014, 2017.

L. Song, S. Vempala, J. Wilmes, and B. Xie. On the complexity of learning neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5514–5522, 2017a.

Z. Song, D. P. Woodruff, and P. Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1611.00898, 2017b.

Z. Song, D. P. Woodruff, and P. Zhong. Relative error tensor low rank approximation. In *SODA*. https://arxiv.org/pdf/1704.08246, 2019.

M. Thorup and Y. Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing*, 41(2):293–331, 2012.

Y. Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. *arXiv preprint arXiv:1703.00560*, 2017.

B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8010–8020, 2018.

D. Van Veen, A. Jalal, E. Price, S. Vishwanath, and A. G. Dimakis. Compressed sensing with deep image prior and learned regularization. *arXiv preprint arXiv:1806.06438*, 2018.

S. Wang, A. Gittens, and M. W. Mahoney. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. *ICML*, 2017.

T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning (ICML)*. arXiv preprint arXiv:1804.09699, 2018.

D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

C. Xiao, P. Zhong, and C. Zheng. Bourgan: Generative networks with metric embeddings. In *NeurIPS*. arXiv preprint arXiv:1805.07674, 2018.

W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Y. Zhang, J. D. Lee, M. J. Wainwright, and M. I. Jordan. Learning halfspaces and neural networks with random initialization. *arXiv preprint arXiv:1511.07948*, 2015.

K. Zhong, Z. Song, and I. S. Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017a.

K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, 2017b.