Enhanced Expressive Power and Fast Training of Neural Networks by Random Projections

Jian-Feng Cai[1,*], Dong Li[1], Jiaze Sun[2], and Ke Wang[1]

[1] *Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, P.R. China.*
[2] *Department of Electrical and Electronic Engineering, Imperial College London, South Kensington Campus, London SW7 2AZ, UK.*

Abstract. Random projections are able to perform dimension reduction efficiently for datasets with nonlinear low-dimensional structures. One well-known example is that random matrices embed sparse vectors into a low-dimensional subspace nearly isometrically, known as the restricted isometric property in compressed sensing. In this paper, we explore some applications of random projections in deep neural networks. We provide the expressive power of fully connected neural networks when the input data are sparse vectors or form a low-dimensional smooth manifold. We prove that the number of neurons required for approximating a Lipschitz function with a prescribed precision depends on the sparsity or the dimension of the manifold and weakly on the dimension of the input vector. The key in our proof is that random projections embed stably the set of sparse vectors or a low-dimensional smooth manifold into a low-dimensional subspace. Based on this fact, we also propose some new neural network models, where at each layer the input is first projected onto a low-dimensional subspace by a random projection and then the standard linear connection and non-linear activation are applied. In this way, the number of parameters in neural networks is significantly reduced, and therefore the training of neural networks can be accelerated without too much performance loss.

1 Introduction

Over the past few years, learning via multiple-layer neural network has been widely studied and has achieved unprecedented success. It has many important applications in image recognition, speech recognition, and natural language processing.

One of the fundamental theoretical question in deep learning is the expressive power of a neural network, which describes its ability to approximate functions. The celebrated universal approximation theorem, which was proved by Cybenko [10], Hornick [17] et

---

*Corresponding author. *Email addresses:* `jfcai@ust.hk` (J.-F. Cai), `madli@ust.hk` (D. Li), `j.sun19@imperial.ac.uk` (J. Sun), `kewang@ust.hk` (K. Wang)

1

al, Funahashi [15] and Barron [3], states that sufficiently large shallow (that is, depth-2 or equivalently, one hidden layer) neural networks can approximate any continuous function on a bounded domain to arbitrary accuracy. But, with a fixed approximation accuracy, the required size of such networks can be exponentially increasing with respect to the dimension. Indeed, Eldan and Shamir [11] proved that there is a continuous function expressed by a small depth-3 feedforward neural networks which cannot be approximated by any shallow network to more than a certain constant accuracy, unless its width grows exponentially in the dimension. This shows the power of depth for feedforward neural network. Lu-Pu-Wang-Hu-Wang [19] studied the expressive power of neural networks from the width point of view. They shown that there exists a class of width-$O(k^2)$ shallow ReLU network that cannot be approximated by any width-$O(k^{1.5})$ and depth-k neural network.

However, the data input in the real world applications are usually structured. For example, images modelled as piecewise smooth functions can have sparse representations under certain orthonormal bases or frames [21]. This means that the intrinsic dimension of the input data is significantly smaller than the ambient space dimension. This fact is often ignored in aforementioned classical approximation results. The expressive power of a neural network may be improved by exploring the structure of the input data. In this direction, Shaham-Cloninger-Coifman [31] studied approximations of functions on a smooth k-dimensional submanifold embedded in $\mathbb{R}^d$. They constructed a depth-4 network and controlled the error of its approximation, where the size of their network depends on k but just weakly on d. Chui-Lin-Zhou [9] studied the expressive power of neural networks in the regression setting when the samples are located approximately on some unknown manifold. They showed that the error of the approximation of their trained depth-3 neural network to the regression function depends on the number of samples, and the dimension of the manifold instead of the ambient dimension.

In this paper, we consider a different approach to analyze the theoretical performance of neural networks with structured input data. Based on our analysis, we propose a new architecture of neural networks, for which the training can be significantly accelerated compared to conventional fully connected or convolutional neural networks. Our main idea is to use linear random projections developed in compressed sensing [14].

For simplicity, we assume that the input data are sparse vectors, namely, k-sparse vectors in $\mathbb{R}^d$. Using the theory of compressed sensing, one can construct a random projection onto an $O(k \log(d/k))$-dimensional space that satisfies the so-called restricted isometry property (RIP), saying that the random projection is nearly isometric when restricted to sparse vectors. Therefore, to get an efficient function approximation, we can first compress the sparse high-dimensional input vectors to low-dimensional ones without changing the metric too much, and then apply a standard neural network with low-dimensional vectors as inputs. In this way, the expressive power is the same as that of the neural network with $O(k \log(d/k))$-dimensional input. In other words, we obtain neural networks with expressive power depending on k and weakly on d. Contrary to the work in [9,31], the neural networks we constructed can be very deep with a fixed width. Our

approach works not only for sparse inputs but also for a large class of structures of input data, for example, when the input data are sampled from a low dimensional manifold.

The idea of using random projection can also be incorporated into fully connected and convolutional neural networks to get new architectures for multi-layer neural networks. At each layer, we first apply a random projection to capture the intrinsic data structure, and then the standard linear transform and nonlinear activation follow. This will improve the overall computational efficiency of the neural networks, since the number of parameters are significantly reduced. In particular, compared to standard fully connected and convolutional neural networks, the training is accelerated drastically. We will also demonstrate the new neural network can achieve comparable accuracy to the original ones.

There is a body of literature (e.g. [18, 30]) regarding neural networks with random weights and the implication for the number of trainable parameters. Compared to those works, our main contribution is to provide a unified framework to improve the approximation power of neural networks for functions on low-dimensional objects. The introduction of random projection in our construction simplifies the proofs significantly and is applicable to various low-dimensional objects. As long as the random projection satisfies RIP on the k-dimensional object, our argument go through and the approximation power of neural networks depends on k and weakly on d.

The rest of this paper is organized as follows. In Section 2, we give a brief introduction to random projections and their stable embedding. In Section 3, we present our theoretical results on the expressive power of neural networks for structured input data. In Section 4, we incorporate random projections into each layer of full connected and convolutional neural networks to get better architecture of deep neural networks. Section 5 is devoted to numerical experiments demonstrating the efficiency and effectiveness of the resulting neural networks.

2   Random Projections and the Restrictive Isometry Property

We will use random projections to study the expressive power and accelerate the training of neural networks. Random projections are able to embed datasets with a non-linear low-dimensional structure into a low-dimensional subspace while almost keeping the metric. In this section, we give a review on related results.

2.1   Compressed Sensing and RIP

Compressed sensing [7, 14] is a signal processing technique that enables acquiring compressible signals from a much smaller number of linear samples than the ambient dimension of signals. It has numerous applications in imaging [7, 20, 27]. Compressed sensing takes advantage of the fact that most signals of interest in practice are compressible: there are only a few nonzero or big elements when the signals are represented over a certain dictionary such as wavelet basis. The key concept in compressed sensing is the restricted

isometric property (RIP), under which many algorithms are able to reconstruct the underlying signal stably. One of the most powerful results in compressed sensing is that some family of random matrices with very few rows will satisfy RIP with high probability.

A signal in $x \in \mathbb{R}^d$ is compressible if $\|Wx\|_0 \le k$ for some $k \ll d$ for some linear transform $W \in \mathbb{R}^{m \times d}$. Here the ℓ_0-norm $\|\cdot\|_0$ stands for the number of nonzeros of a vector. We also call such an x a k-sparse signal. In other words, a signal is compressible if it is sparse under certain linear transform. This assumption holds true for a wide variety of classes of signals. For example, piecewise smooth signals are (nearly) sparse under the representation of orthogonal wavelets or wavelet frames such as the curvelet [21]. Actually, the sparsity assumption is the foundation of many models and approaches in modern signal processing and imaging.

Since the degree of freedom in a k-sparse signal is only k with $k \ll d$, it is possible to acquire the signal efficiently by n linear samples with $n \ll d$. This is exploited by compressed sensing. In the encoding stage of compressed sensing, we acquire a k-sparse sparse x by $b := Ax$, where $A \in \mathbb{R}^{n \times d}$ is a sampling matrix with each row corresponding to one linear sample. Compared to traditional signal processing where the full sample of x is required, compressed sensing can save sampling costs significantly and can be applied to a wider range of imaging scenarios for which the full sampling is prohibited. In the decoding stage, one wants to recover the k-sparse signal x from b. Various approaches are available with recovery performance guarantee, including convex optimization based approaches [7] and non-convex ones [4, 13, 26]; see also the book [14] and the references therein.

A key concept in compressed sensing theory is the restricted isometric property (RIP) introduced by E. Candès and T. Tao [8]. For simplicity, we assume the sparse transform $W = I$, i.e., the underlying signal x satisfies

$$x \in \mathcal{S}_k := \{y \in \mathbb{R}^d \, : \, \|y\|_0 \le k\}.$$

Then, to have a successful recovery of any signal $x \in \mathcal{S}_k$ from $b = Ax$, we should at least require that the sampling operator A is injective on $\mathcal{S}_k$. That is, it is necessary $\|A(x_1 - x_2)\|_2 > 0$ for any $x_1, x_2 \in \mathcal{S}_k$ satisfying $x_1 \ne x_2$. However, there will always be noise in the practical measurements, and also the injectivity may be too weak to design recovery algorithms. To have a stable recovery with practical algorithms, we need to strength the restricted injectivity to the following restricted isometric property (RIP):

$$(1 - \delta_s)\|y\|_2 \le \|Ay\|_2 \le (1 + \delta_s)\|y\|_2 \quad \text{for all } y \in \mathcal{S}_s, \tag{2.1}$$

where $\delta_s \in (0, 1)$. Here we assume A is normalized so that its restricted eigenvalues are concentrated around 1. If we choose $s = 2k$, then RIP (2.1) implies that, for any k-sparse vectors x_1 and x_2, $\|A(x_1 - x_2)\|_2$ is almost the same as $\|x_1 - x_2\|_2$. In other words, if A satisfies RIP, then the application of A preserves the metric of the set $\mathcal{S}_k$ inherited from $\mathbb{R}^d$. This enables us to design efficient and stable algorithms for the recovery of sparse signals with a theoretical performance guarantee [4,7,13,26]. For example, it was shown [5] that if

A satisfies RIP with $\delta_{2k} < \sqrt{2} - 1$, then the solution of the following ℓ_1-norm minimization gives a stable recovery of the k-sparse vector x

$$\min_{y \in \mathbb{R}^d} \|y\|_1 \quad \text{s.t.} \quad \|Ay - b\|_2 \leq \sigma, \tag{2.2}$$

where $b = Ax + \epsilon$ are the noisy linear samples of x with the noise ϵ satisfying $\|\epsilon\|_2 \leq \sigma$. Actually, even when x is not exactly in $\mathcal{S}_k$ but only close to it, (2.2) is still able to give a faithful recovery of x. The main ingredient in (2.2) is the minimization of the ℓ_1-norm function, which is the tightest convex approximation of the ℓ_0-norm function. Therefore, (2.2) finds somewhat a solution with the smallest ℓ_0-norm in the feasible set, which is close to x.

2.2 Random Projections Satisfy RIP

An important question is then to find matrices $A \in \mathbb{R}^{n \times d}$ with good RIP constants using the smallest possible n. Since computing the RIP constants is strongly NP-hard, it is very difficult to use numerical methods to construct RIP matrices. Also, any existing deterministic matrices satisfying RIP will not have an optimal n. The best known deterministic RIP matrices have a number of rows $n \geq O(s^2)$, and it is still an open problem to construct a deterministic matrix satisfying RIP with an optimal n. A major breakthrough in compressive sensing is the use of random matrices to construct RIP matrices with optimal n. In particular, let $A \in \mathbb{R}^{n \times d}$ be a random matrix whose entries are independent Gaussian random variables with mean 0 and variance $1/n$. Then, with overwhelming probability, A satisfies RIP with constant $\delta_s > 0$ provided $n \geq O(\delta_s^{-2} s \log(d/s))$. This result is summarized in the following theorem.

Theorem 2.1 (Theorem 9.2 in [14]). *Let $A \in \mathbb{R}^{n \times d}$ be a random matrix whose entries are independent Gaussians with mean-0 variance-$1/n$. Then, there exists a universal constant $C > 0$ such that A satisfies RIP (2.1) with constant $\delta_s \in (0, \delta)$ with probability at least $1 - 2e^{-\frac{\delta^2}{2C}n}$ provided*

$$n \geq 2C\delta^{-2} s \log(ed/s).$$

The theorem was first proved in [8] with a weaker bound in a more restrictive setting, and simple proofs can be found in [1, 14]. Besides random Gaussian matrices, there exist other types of random matrices satisfying RIP with $n \sim O(s \log^\alpha(d))$ for some $\alpha > 0$, such as subGaussian, Bernoulli, and random rows of discrete Fourier transform matrices [1, 14, 29].

The RIP above can be extended to general cases where the sparsifying transform W is not necessarily the identity. When $W \in \mathbb{R}^{d \times d}$ is orthogonal, RIP (2.1) can be adapted to

$$(1 - \delta_s)\|y\|_2 \leq \|Ay\|_2 \leq (1 + \delta_s)\|y\|_2 \quad \text{for all } y \in \mathcal{S}_s^W, \tag{2.3}$$

for some $\delta_s \in (0, 1)$, where

$$\mathcal{S}_s^W = \{y \in \mathbb{R}^d \ : \ \|Wy\|_0 \leq k\}.$$

Since Gaussian random variables are unitary invariant, a simple calculation and Theorem 2.1 give that, for any orthogonal W, a Gaussian random matrix $A \in \mathbb{R}^{n \times d}$ satisfies the generalized RIP (2.3) with high probability provided $n \sim O(s\log(d/s))$. When $W \in \mathbb{R}^{m \times d}$ forms a tight frame (i.e., $W^T W = I$), the generalized RIP was studied in [6,28]. Gaussian random matrices satisfy the generalized RIP there with high probability and optimal n.

2.3 RIP on Smooth Manifolds

Besides sparse signal models, there is another common model called manifold signal model. This model generalizes the notion of concise signal structure beyond the framework of bases and representations. It arises in broad cases, for example, where a k-dimensional parameter can be identified that carries the relevant information about a signal that changes as a continuous function of these parameters. In general, this dependence may not be neatly reflected in a sparse set of transform coefficients. In [2], Baraniuk and Wakin proposed a approach for nonadaptive dimensionality reduction of manifold-modeled data, where they demonstrated that a small number of random linear projections can preserve key information about a manifold-modeled signal. To state their results, we need a few definitions for a Riemannian manifold.

Let $\mathcal{M}$ be a k-dimensional compact Riemannian submanifold embedded in $\mathbb{R}^d$. The condition number is defined as $1/\tau$, where τ is the largest number having the following property: for every $r < \tau$, the tubular neighborhood of $\mathcal{M}$ of radius r in $\mathbb{R}^d$ defined as $\{x + \eta \in \mathbb{R}^d : x \in \mathcal{M}, \eta \in \mathrm{Tan}_x^\perp, \|\eta\|_2 < r\}$, where $\mathrm{Tan}_x^\perp$ denotes the set of vectors normal to the tangent space at x, is embedded in $\mathbb{R}^d$. Given $T > 0$, the geodesic covering number $G(T)$ of $\mathcal{M}$ is defined as the smallest number such that there exists a finite set $\mathcal{A} \subset \mathcal{M}$ of $G(T)$ points so that

$$\min_{a \in \mathcal{A}} d_{\mathcal{M}}(a, x) \leq T$$

for all $x \in \mathcal{M}$, where $d_{\mathcal{M}}(a, x)$ is the geodesic distance between a and x. We say that $\mathcal{M}$ has geodesic covering regularity R for resolutions $T \leq T_0$ if

$$G(T) \leq \frac{R^k V k^{k/2}}{T^k},$$

where V is the volume of $\mathcal{M}$.

Theorem 2.2 (Theorem 3.1 in [2]). *Let $\mathcal{M}$ be a compact k-dimensional Riemannian submanifold of $\mathbb{R}^d$ having condition number $1/\tau$, volume V, and geodesic covering regularity R. Fix $0 < \delta < 1$ and $0 < \rho < 1$. Let $A = \sqrt{\frac{d}{n}} \Phi$, where $\Phi \in \mathbb{R}^{n \times d}$ is a random orthoprojector with*

$$n = O\left(\frac{k\log(dVR\tau^{-1}\delta^{-1})\log(1/\rho)}{\delta^2} \right).$$

If $n \leq d$, then with probability at least $1 - \rho$, the following statement holds: For every $x, y \in \mathcal{M}$,

$$(1-\delta)\|x - y\|_2 \leq \|Ax - Ay\|_2 \leq (1+\delta)\|x - y\|_2. \tag{2.4}$$

Hence, if $k \ll d$, then we reduce the dimension of input data from d to $O(k\log d)$. The diameter of the manifold is defined by

$$\text{diam}(M) = \sup_{x,y \in M} d_M(x,y),$$

where $d_M(x,y)$ is the geodesic distance between x,y on M.

3   Improved Expressive Power of Neural Networks by Random Projections

In this section, we use the random projections discussed in the previous section to explore the expressive power of neural networks for functions on datasets with a low-dimensional structure in the ambient space $\mathbb{R}^d$.

3.1   Fully-Connected Neural Networks (FCNN)

There are many artificial neural network architectures available, such as the fully-connected neural network (FCNN), the convolutional neural network (CNN), and the recurrent neural network. We will study the expressive power a multi-layer fully-connected neural network with the rectified linear unit (ReLU) as the activation function.

The ReLU is so far the most popular activation function for deep neural networks, and it is the positive part of its argument. More precisely, let $z \in \mathbb{R}^\ell$, and the ReLU is defined by

$$\text{ReLU}(z) := \begin{bmatrix} \max\{0, z_1\} \\ \vdots \\ \max\{0, z_\ell\} \end{bmatrix}, \quad \forall z \in \mathbb{R}^\ell.$$

Let $x \in \mathbb{R}^d$ be the input. Then it generates outputs $x^{(l)}$ at the l-th layer recursively by: $x^{(0)} = x$, and

$$x^{(l)} = \text{ReLU}\left(W^{(l)} x^{(l-1)}\right), \quad l = 1, 2, \ldots, L,$$

where $W^{(l)} : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ is the affine transformation at layer l. The final output y of the neural network is

$$y = W^{(L+1)} x^{(L)},$$

where $W^{(L+1)} : \mathbb{R}^{d_L} \to \mathbb{R}$ is an affine transformation. Therefore, the function represented by the fully-connected neural network is

$$W^{(L+1)} \circ \text{ReLU} \circ W^{(L)} \circ \cdots \circ \text{ReLU} \circ W^{(1)}.$$

The number L is usually called the depth of the network, and the d_l's are called the widths. The sum $\sum_{l=1}^{L} d_l$ is called the number of neurons of the network. If $L = 1$, then the network is called a shallow neural network. If $L > 1$, it is called a deep neural network.

3.2 Expressive Power of Neural Networks

As mentioned in the introduction, by the celebrated universal approximation theorem, we know that sufficiently large shallow neural networks can approximate any continuous function on a bounded domain to arbitrary accuracy. That is, for a continuous function $f \in C([-1,1]^d)$ and for every $\varepsilon > 0$, there exists a shallow ReLU neural network f_ε such that

$$\|f - f_\varepsilon\|_{L^\infty([-1,1]^d)} := \max_{x \in [-1,1]^d} |f(x) - f_\varepsilon(x)| < \varepsilon.$$

However, the universal approximation theorem does not tell the number of neurons that f_ε has, or equivalently the approximation accuracy. There have been many literatures on studying the number of neurons, or the approximation accuracy, of (either shallow or deep) neural networks since the work [3] by Barron, and we know now that the number of neurons that the neural network f_ε needs will depend on the regularity (e.g., the modulus continuity) of the function f.

Barron [3] first gave a quantitative approximation rate in L^2 norm for shallow neural networks, assuming the function f has bounded first moment of the magnitude of the Fourier transform. If f is r times differentiable, then Mhaskar [23] obtained an optimal quantitative approximation rate in L^2 norm, that is, for every $\varepsilon > 0$, there exists a shallow neural network f_ε with $O(\varepsilon^{-d/r})$ neurons such that

$$\|f - f_\varepsilon\|_{L^2([-1,1]^d)} < \varepsilon.$$

If f is C^2 and has bounded Hessian, then Shaham-Cloninger-Coifman [31] proved the same quantitative approximation rate in L^∞ norm. Their result also holds if f is supported in a lower dimensional manifold. In a recent work [16] of Hanin, several approximation results of ReLU neural networks were obtained for continuous functions, convex functions, and smooth functions, respectively. In particular, if f is bounded and Lipschitz continuous, then Theorem 1 in [16] tells that for every $\varepsilon > 0$, there exists a ReLU neural network with $Cdd!\varepsilon^{-d}$ neurons, where C is a positive constant depending only on the Lipschitz constant of f and is independent of d, such that

$$\|f - f_\varepsilon\|_{L^\infty([-1,1]^d)} < \varepsilon.$$

Recall that a function $f : E \to \mathbb{R}$ is called Lipschitz continuous on the set $E \subset \mathbb{R}^d$ if

$$\mathrm{Lip}(f) := \sup \left\{ \frac{|f(x) - f(y)|}{\|x - y\|_2} \mid x, y \in E, \, x \neq y \right\} < \infty,$$

where $\|x - y\|_2$ is the distance between x and y in $\mathbb{R}^d$. If f is Lipschitz continuous, then $\mathrm{Lip}(f)$ defined above is called the Lipschitz constant. Note that Theorem 1 in [16] is stated for positive functions, but it is clearly true for bounded function as well by subtracting a large constant from the neural network.

3.3   Improved Expressive Power for Functions with Sparse Inputs

The aforementioned results on expressive power assumes that the domain of functions is $[-1,1]^d$. However, in real applications, the input data x is often structured. For example, in an image recognition task, the input data are digital images with d pixels, which obviously are not arbitrary in $[-1,1]^d$. By considering the structure of the input data, we expect to obtain better results on the expressive power of neural networks than results reviewed in the previous section, which ignore structures of input data.

There are several approximation results taking into account the structure of the input. Shaham-Cloninger-Coifman [31] assumed that the domain of the function is a k-dimensional smooth submanifold embedded in $\mathbb{R}^d$ and proved that the size of their constructed approximating networks depends on k but just weakly on d. In [9], Chui-Lin-Zhou used neural networks to approximate regression functions, where the training samples are located approximately on some unknown manifold. It was shown there that the error of the approximation of their trained depth-3 neural network to the regression function depends on the number of samples and the dimension of the manifold instead of the ambient dimension. In both of these two papers, multi-layer neural networks are constructed, for which each hidden layer is endowed with a specific learning task.

Here we provide an improved expressive power by assuming the sparsity of the input vectors and using random projections. Our assumption is motivated by the facts that many learning tasks are with images or audio signals as inputs and that images and audio signals have sparse representation under suitable basis. For simplicity, we assume the input vectors are in $\mathcal{S}_k = \{y \in \mathbb{R}^d : \|y\|_0 \le k\}$ with $k \ll d$, and the extension to the sparse case under a linear transformation (i.e., the input vectors are in $\mathcal{S}_s^W$) is straightforward.

We will show in the below that, for Lipschitz continuous functions that are defined on $\mathcal{S}_k$, we can choose a neural network with $Cnn!\varepsilon^{-n}$ neurons, where $n = O(k\log(d/k))$ and C is a positive constant depending only on the Lipschitz constant of f, to approximate f with accuracy ε. Our proof will make use of a theorem of McShane [22] and Whitney [33] on the extension of Lipschitz functions, which states that any Lipschitz continuous function defined on an arbitrary subset of $\mathbb{R}^n$ can be extended to be a Lipschitz continuous function in $\mathbb{R}^n$ with the same Lipschitz constant. (See also Theorem 3.1 in the book [12] of Evans and Gariepy.)

Theorem 3.1. *Let $\mathcal{S}_k = \{y \in \mathbb{R}^d : \|y\|_0 \le k\}$ with $k \ll d$. Suppose $f : \mathcal{S}_k \to \mathbb{R}$ is a Lipschitz continuous function with Lipschitz constant* $\mathrm{Lip}(f)$, *that is,*

$$\mathrm{Lip}(f) = \sup_{x,y \in \mathcal{S}_k,\ x \ne y} \frac{|f(x) - f(y)|}{\|x - y\|_2} < \infty.$$

Then for sufficiently large m,

$$\inf_{f_0 \in F_m^d} \sup_{x \in \mathcal{S}_k \cap [-1,1]^d} |f(x) - f_0(x)| \le C\mathrm{Lip}(f) k^{\frac{3}{2}} \log(d/k) m^{-\frac{C}{k\log(d/k)}},$$

where F_m^d is the set of functions represented by ReLU fully-connected neural networks with m neurons and d inputs, and C is a universal positive constant.

A few remarks are as follows.

Remark 3.1. Theorem 3.1 estimates the L^∞ norm between f and its neural network approximations, rather than the L^p norm (with $p < \infty$). It is more natural (at least to us) to use the L^∞ norm when considering approximating functions defined on a low dimensional set of an ambient space.

Remark 3.2. The Lipschitz continuity assumption of f ensures that we can extend f, which is defined on $\mathcal{S}_k$, to a Lipschitz continuous function on $\mathbb{R}^d$ with the same Lipschitz constant as that of f.

Remark 3.3. In applications, the sparsity is usually much smaller than the dimension, i.e., $k \ll d$. Then in order to have

$$\inf_{f_0 \in F_m^d} \sup_{x \in \mathcal{S}_k \cap [-1,1]^d} |f(x) - f_0(x)| \le \varepsilon,$$

it suffices to require that $\log m = 2k(\log k + \log\log d - \frac{1}{2}\log\varepsilon) \cdot \log d$, i.e.,

$$m = d^{2k(\log k + \log\log d) - k\log\varepsilon},$$

which is significantly smaller than exponential functions of d.

Proof of Theorem 3.1. Fix a $\delta \in (0,1)$, say $\delta = 1/2$. Theorem 2.1 implies that there exists a matrix $A \in \mathbb{R}^{n \times d}$, where $n = C_0 k \log(d/k)$ with a universal positive constant C_0, satisfying the restricted isometric property (2.1) with constant $\delta = 1/2$. Since A satisfies RIP, the map $x \mapsto Ax$ is a bijection from $\mathcal{S}_k$ to $\Omega := \{Ax \mid x \in \mathcal{S}_k\}$. For every $y \in \Omega$, define

$$g(y) = f(x),$$

where x is the unique element in $\mathcal{S}_k$ such that $Ax = y$.

Since f is a Lipschitz continuous on $\mathcal{S}_k$ with Lipschitz constant $\mathrm{Lip}(f)$ and A satisfies RIP, it is elementary to show that g is Lipschitz continuous on Ω with Lipschitz constant at most $2\mathrm{Lip}(f)$. Notice that $\Omega \subseteq \mathbb{R}^n$. Then by the theorem of McShane [22] and Whitney [33], we can extend g to be a Lipschitz continuous function in $\mathbb{R}^n$ with the same Lipschitz constant.

For every $x \in \mathcal{S}_k \cap [-1,1]^d$, we have

$$\|Ax\|_\infty \le \|Ax\|_2 \le \frac{3}{2}\|x\|_2 \le \frac{3\sqrt{k}}{2}.$$

Hence,

$$\{Ax \mid x \in \mathcal{S}_k \cap [-1,1]^d\} \subset \left[-\frac{3\sqrt{k}}{2}, \frac{3\sqrt{k}}{2}\right]^n.$$

Now we consider

$$\tilde{g}(\boldsymbol{y}) = g\left(\frac{3\sqrt{k}}{2}(2\boldsymbol{y}-\boldsymbol{1})\right),$$

where $\boldsymbol{1} = (1,\cdots,1)^T \in \mathbb{R}^n$. The function $\tilde{g}$ is Lipschitz continuous on $\mathbb{R}^n$ with Lipschitz constant $\mathrm{Lip}(\tilde{g}) = 3\sqrt{k}\mathrm{Lip}(g) \leq 6\sqrt{k}\mathrm{Lip}(f)$. By Theorem 1 in [16], we have that for every $\varepsilon > 0$, there exists a ReLU neural network g_ε with at most $m = C_1 n n! (6\sqrt{k}\mathrm{Lip}(f))^n \varepsilon^{-n}$ neurons such that

$$\|\tilde{g} - g_\varepsilon\|_{L^\infty([0,1]^n)} \leq \varepsilon,$$

where $C_1 > 0$ is a universal constant. In terms of f, we obtain

$$\inf_{h \in F_m^n} \sup_{\boldsymbol{x} \in \mathcal{S}_k \cap [-1,1]^d} |f(\boldsymbol{x}) - h(\boldsymbol{A}\boldsymbol{x})| \leq C_2 \sqrt{k}\mathrm{Lip}(f) n m^{-\frac{1}{n}},$$

where $C_2 > 0$ is an another universal constant. Let $f_0 = h(\boldsymbol{A}\cdot)$. Since $h \in F_m^n$, we can rewrite it as $h = \boldsymbol{W}^{(L+1)} \circ \mathrm{ReLU} \circ \boldsymbol{W}^{(L)} \circ \cdots \circ \mathrm{ReLU} \circ \boldsymbol{W}^{(1)}$. Therefore, $f_0 = \boldsymbol{W}^{(L+1)} \circ \mathrm{ReLU} \circ \boldsymbol{W}^{(L)} \circ \cdots \circ \mathrm{ReLU} \circ (\boldsymbol{W}^{(1)} \circ \boldsymbol{A}) \in F_m^d$, which concludes the proof. $\square$

3.4 Improved Expressive Power for Functions on Smooth Manifolds

Using similar techniques, we can also improve the expressive power of FCNN with inputs from a compact k-dimensional Riemannian manifold embedded in $\mathbb{R}^d$. We assume that the underlying function is Lipschitz continuous, which is weaker than that in [31]. It turns out that the number of neurons required for an ε-approximation in the infinity-norm depends weakly on d. Our main tool in the proof is Theorem 2.2, which states that random projections are stable embeddings.

Theorem 3.2. *Let $(\mathcal{M},g)$ be a compact k-dimensional Riemannian submanifold of $\mathbb{R}^d$. Let $f : \mathcal{M} \to \mathbb{R}$ be a Lipschitz continuous function with Lipschitz constant $\mathrm{Lip}(f)$, that is,*

$$\mathrm{Lip}(f) = \sup_{\boldsymbol{x},\boldsymbol{y} \in \mathcal{M},\, \boldsymbol{x} \neq \boldsymbol{y}} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_2} < \infty.$$

Then for sufficiently large m,

$$\inf_{f_0 \in F_m^d} \sup_{\boldsymbol{x} \in \mathcal{M}} |f(\boldsymbol{x}) - f_0(\boldsymbol{x})| \leq C\mathrm{Lip}(f)\mathrm{diam}(M)\log(2dVR\tau^{-1})m^{-\frac{C}{\log(2dVR\tau^{-1})}},$$

where F_m^d is the set of functions represented by ReLU fully-connected neural networks with m neurons and d inputs, and τ is the condition number of $\mathcal{M}$ in $\mathbb{R}^d$, V is the volume of $(\mathcal{M},g)$, R is the geodesic covering regularity of $\mathcal{M}$, $\mathrm{diam}(M)$ is the diameter of $(\mathcal{M},g)$ and C is a universal positive constant.

Proof. Fix a $\delta \in (0,1)$, say $\delta = 1/2$. Theorem 2.2 implies that there exists a matrix $A \in \mathbb{R}^{n \times d}$ satisfies the restricted isometric property (2.4) with constant $\delta = \frac{1}{2}$. Here, $n = C_0 k \log(2dVR\tau^{-1})$ with $C_0 > 0$ is a universal constant. Since A satisfies RIP, the map $x \mapsto Ax$ is a bijection from $\mathcal{M}$ to $\Omega := \{Ax \mid x \in \mathcal{M}\} \subset \mathbb{R}^n$. For every $y \in \Omega$, define

$$\phi(y) = f(x),$$

where x is the unique element in $\mathcal{M}$ such that $Ax = y$. As in the proof of Theorem 3.1, ϕ is Lipschitz continuous on Ω with Lipschitz constant at most $2\text{Lip}(f)$ and can be extended to be a Lipschitz continuous function in $\mathbb{R}^n$ with the same Lipschitz constant.

For every $x_1, x_2 \in \mathcal{M}$,

$$\|Ax_1 - Ax_2\|_\infty \leq \|Ax_1 - Ax_2\|_2 \leq \frac{3}{2}\|x_1 - x_2\|_2 \leq \frac{3}{2}\text{diam}(M).$$

Now we consider

$$\tilde{\phi}(y) = \phi\left(\frac{3}{2}\text{diam}(M)y + y_0\right),$$

where $y_0 \in \mathbb{R}^n$ is chosen such that $\{\frac{3}{2}\text{diam}(M)y + y_0, y \in [0,1]^n\} \subset \Omega$. Then $\tilde{\phi}$ is Lipschitz continuous on $\mathbb{R}^n$ with Lipschitz constant $\text{Lip}(\tilde{\phi}) = \frac{3}{2}\text{diam}(M)\text{Lip}(\phi) = 3\text{diam}(M)\text{Lip}(f)$. By Theorem 1 in [16], we have that for every $\varepsilon > 0$, there exists a ReLU neural network ϕ_ε with at most $m = C_1 n n! (3\text{diam}(M)\text{Lip}(f))^n \varepsilon^{-n}$ neurons such that

$$\|\tilde{\phi} - \phi_\varepsilon\|_{L^\infty([0,1]^n)} \leq \varepsilon,$$

where $C_1 > 0$ is a universal constant. In terms of f, we obtain

$$\inf_{h \in F_m^n} \sup_{x \in \mathcal{M}} |f(x) - h(Ax)| \leq C_2 \text{Lip}(f)\text{diam}(M)nm^{-\frac{1}{n}},$$

where $C_2 > 0$ is an another universal constant. Now we can conclude the proof in the same way as in the proof of Theorem 3.1. $\square$

Remark 3.4. The volume of $\mathcal{M}$ and the geodesic covering regularity of $\mathcal{M}$ do not depend on the ambient space $\mathbb{R}^d$, and hence, do not depend on d. In the definition of the condition number of $\mathcal{M}$ in $\mathbb{R}^d$, it appears to depend on $\mathbb{R}^d$. However, the dependence on d is very weak. For example, by the Nash embedding theorem [25], $\mathcal{M}$ can be isometrically embedded in $\mathbb{R}^{k(3k+11)/2}$. If $d \geq \frac{k(3k+11)}{2}$, then the condition number of $\mathcal{M}$ in $\mathbb{R}^d$ is the same as the condition number of $\mathcal{M}$ in $\mathbb{R}^{k(3k+11)/2}$.

Remark 3.5. There are examples shown in [24, 32] that one pixel change will make deep neural networks misclassify natural images. Such changes induce a severe jump in the Lipschitz constant (which is defined with respect to the Euclidean distance as in Theorem 3.2). From the practical point of view, the natural distance of the input data should perhaps be the Euclidean distance rather than the geodesic distance on the manifold. That is part of the reason why the Lipschitz continuity of f in Theorem 3.2 is stated in terms of the Euclidean distance $\|\cdot\|_2$.

4 Accelerate the Training of Neural Networks by Random Projections

In this section, we use the random projections to accelerate the training of neural networks. We present in detail our implementation of random projections in both fully-connected and convolutional neural networks. We shall also provide estimates to demonstrate that our scheme indeed achieves significant reduction in computational complexity and number of parameters.

4.1 Fully-connected Neural Networks (FCNN)

The main idea in the proof of Theorems 3.1 and 3.2 is to use random projections to reduce the number of neurons. In practice, we will also use random projections to reduce the number of neurons and hence the number of parameters of FCNN. Consequently, the training of FCNN is significantly accelerated.

Here for simplicity, we assume that the input vectors are sparse under a suitable linear transformation, which is a common assumption for digital images and signals. It is similar when input signals are on a smooth manifold. The bulk of the computations of an FCNN comes from the matrix multiplication in each layer. Recall that a multi-layer FCNN produces outputs by

$$x^{(l)} = \text{ReLU}\left(W^{(l)}x^{(l-1)}\right), \quad \text{for } l=1,\dots,L, \tag{4.1}$$

where $W^{(l)} : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_{l-1}}$ is an affine transformation, with d_{l-1} and d_l respectively being the dimensions of the input $x^{(l-1)}$ and output $x^{(l)}$. The final output is $y = W^{(L+1)}x^{(L)}$ with $W^{(L+1)} : \mathbb{R}^{d_L} \to \mathbb{R}$ an affine transformation.

We modify (4.1) by random projection as in the following. At layer-1, the input vector $x^{(0)} = x \in \mathcal{S}_k$ is sparse. According to Theorem 2.1, if we choose a suitable n_0, then with high probability a Gaussian random matrix $A^{(1)} \in \mathbb{R}^{n_0 \times d_0}$ embeds $\mathcal{S}_k$ to $\mathbb{R}^{n_0}$ nearly isometrically. Therefore, $A^{(1)}$ reduce the dimension of $\mathcal{S}_k$ without too much information loss. We then apply an affine transformation $U^{(1)} : \mathbb{R}^{n_0} \to \mathbb{R}^{d_1}$ on the embedded subspace $\mathbb{R}^{n_0}$. In other words, we replace $W^{(1)}$ by $U^{(1)} \circ A^{(1)}$. In this way, there are only $d_1(n_0+1)$ parameters in layer-1, which is significantly smaller than $d_1(d_0+1)$ in (4.1). Since ReLU set negative entries to 0, the outputs of each layer are sparse vectors. The same parameter reduction procedure as in layer-1 is applied to each layer. In particular, at layer-l, we replace $W^{(l)}$ by $U^{(l)} \circ A^{(l)}$, where $U^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{d_l}$ is an affine transformation to be trained and $A^{(l)} \in \mathbb{R}^{n_{l-1} \times d_{l-1}}$ is a given matrix drawn from random Gaussian distribution. The number of parameters at layer-l is reduced from $d_l(d_{l-1}+1)$ in (4.1) to $d_l(n_{l-1}+1)$. Altogether, we propose the following FCNN

$$x^{(l)} = \text{ReLU}\left(U^{(l)} \circ A^{(l)}x^{(l-1)}\right), \quad \text{for } l=1,\dots,L, \tag{4.2}$$

where $\boldsymbol{U}^{(l)}:\mathbb{R}^{n_{l-1}}\to\mathbb{R}^{d_l}$ is an affine transformation to be trained from the data, and $\boldsymbol{A}^{(l)}\in\mathbb{R}^{n_{l-1}\times d_{l-1}}$ is a fixed matrix with entries drawn from i.i.d. random Gaussian distribution. The final output is $y=\boldsymbol{W}^{(L+1)}\boldsymbol{x}^{(L)}$ with $\boldsymbol{W}^{(L+1)}:\mathbb{R}^L\to\mathbb{R}$ an affine transformation to be trained.

Under this scheme, the number of parameters can be significantly reduced so long as n_l's are kept small. In fact, it can be immediately calculated that the number of parameters of the original FCNN (4.1) is $(d_L+1)+\sum_{l=1}^L(d_{l-1}+1)d_l$, whereas that of the modified network (4.2) is $(d_L+1)+\sum_{l=1}^L(n_{l-1}+1)d_l$. In practice, since it is usually the case that d_1 and d_{l-1} are large and the input $\boldsymbol{x}^{(l)}$ is sparse, $n_{l-1}\ll d_{l-1}$ can be easily satisfied. Therefore, the reduction of number of parameters by (4.2) is significant.

4.2   Convolutional Neural Networks (CNN)

Since convolution is linear in nature, it is possible to adapt our random projection scheme from FCNNs to CNNs. Again for simplicity, consider the l-th layer of a CNN with a ReLU activation:

$$\mathcal{X}^{(l)}_{\cdot\cdot j}=\text{ReLU}\left(\mathcal{F}^{(l)}_{\cdot\cdot\cdot j}*\mathcal{X}^{(l-1)}\right),\quad 1\le j\le c_l. \tag{4.3}$$

Here for $i=l-1$ or l, $\mathcal{X}^{(i)}\in\mathbb{R}^{m\times m\times c_i}$ is the output tensor at the i-th layer, with height and width m, and c_i number of channels; $\mathcal{F}^{(l)}\in\mathbb{R}^{h\times h\times c_{l-1}\times c_l}$ is a trainable tensor consisting of c_l filters of height and width h and depth c_{l-1}. In addition, the "$\cdot$" notation in the subscripts means all entries in that axis are included, for instance $\mathcal{X}^{(l)}_{\cdot\cdot j}$ is simply the $m\times m$ matrix at the j-th channel of the tensor $\mathcal{X}^{(l)}$, and $\mathcal{F}^{(l)}_{\cdot\cdot\cdot j}$ is the j-th filter at the l-th layer. We shall describe two possible modifications of the CNN by random projections.

4.2.1   Approach I: Direct Extension

Upon realizing that convolution is essentially a matrix multiplication acting on different patches of the input tensor, (4.3) can be rewritten more succinctly as

$$\boldsymbol{X}^{(l)}=\text{ReLU}\left(\boldsymbol{F}^{(l)}\tilde{\boldsymbol{X}}^{(l-1)}\right), \tag{4.4}$$

where $\tilde{\boldsymbol{X}}^{(l-1)}\in\mathbb{R}^{c_{l-1}h^2\times m^2}$ is a matrix whose columns are vectorized $h\times h\times c_{l-1}$ patches of $\mathcal{X}^{(l-1)}$ to be convolved with the filters; $\boldsymbol{F}^{(l)}\in\mathbb{R}^{c_l\times c_{l-1}h^2}$ is a matrix wherein each row is given by $\boldsymbol{F}^{(l)}_{j\cdot}=\text{vec}(\mathcal{F}^{(l)}_{\cdot\cdot\cdot j})^T$; $\boldsymbol{X}^{(l)}\in\mathbb{R}^{c_l\times m^2}$ is the matrix whose rows are vectorized channels of $\mathcal{X}^{(l)}$, namely $\boldsymbol{X}^{(l)}_{j\cdot}=\text{vec}(\mathcal{X}^{(l)}_{\cdot\cdot j})^T$. As argued in the previous section, columns of $\tilde{\boldsymbol{X}}^{(l-1)}$ are sparse. Therefore, following (4.2), instead of $\boldsymbol{F}^{(l)}$, we first do dimension reduction of the sparse vectors by a random Gaussian matrix $\boldsymbol{A}^{(l)}$, followed by a linear transformation $\boldsymbol{U}^{(l)}$ in the reduced subspace. We obtain

$$\boldsymbol{X}^{(l)}=\text{ReLU}\left(\boldsymbol{U}^{(l)}\boldsymbol{A}^{(l)}\tilde{\boldsymbol{X}}^{(l-1)}\right), \tag{4.5}$$

where, similar to the FCNN scheme, $U^{(l)} \in \mathbb{R}^{c_l \times n_{l-1}}$ is trainable, and $A^{(l)} \in \mathbb{R}^{n_{l-1} \times c_{l-1} h^2}$ is a fixed matrix whose entries are drawn from i.i.d. random Gaussian distribution. The effect of the larger variance of $A^{(l)}$ can be nullified by batch normalization. Similar to the FCNN case, the number of parameters has been reduced because of the approximation $F^{(l)}$ by $U^{(l)} A^{(l)}$ with a small n_{l-1}.

4.2.2 Approach II: Per-channel Extension

In the first approach, columns in $A^{(l)}$ that correspond to one channel of the input tensor are independent from those corresponding to another. In other words, different channels of the input tensor are essentially assigned different "A"s. This means that the scheme does not take into account the correlations amongst the input channels, however in practice, for instance, the RGB channels of an input image are closely related. As a result, this might diminish the expressive power of the network. To remedy this, we can consider apply random projections separately for each channel, and then perform a summation over all the per-channel outputs. Let $\hat{X}^{(l-1)} \in \mathbb{R}^{h^2 \times m^2 \times c_{l-1}}$ be the tensor such that the columns of $\hat{X}^{(l-1)}_{\cdot \cdot j}$ are vectorized $h \times h$ patches from the j-th channel of $\mathcal{X}^{(l-1)}$. Then our modified scheme can be written as

$$X^{(l)} = \text{ReLU}\left(\sum_{j=1}^{c_{l-1}} \hat{U}^{(l)}_{\cdot \cdot j} \hat{A}^{(l)} \hat{X}^{(l-1)}_{\cdot \cdot j}\right), \tag{4.6}$$

where $\hat{U}^{(l)} \in \mathbb{R}^{c_l \times n_{l-1} \times c_{l-1}}$ is trainable, and $\hat{A}^{(l)} \in \mathbb{R}^{n_{l-1} \times h^2}$ is a given matrix whose entries are randomly drawn from i.i.d. Gaussian distribution.

A small price for this approach is an increase in computational and model complexity compared to the first approach (4.5), but it still achieves substantial, albeit less dramatic, reduction over the original CNN (4.3) as long as n_{l-1} is small.

5 Experiments

In this section, we demonstrate that our schemes in Section 4 indeed achieve significant reduction in model and computational complexity while causing minimal loss in classification accuracy.

We used two well-known data sets, MNIST and CIFAR-10. MNIST is a collection of 28×28 images of handwritten digits from 0 to 9, in which there are 60,000 training examples 10,000 testing ones. CIFAR-10 consists of $32 \times 32 \times 3$ images, where 3 indicates the three RGB channels, and there are 50,000 training examples and 10,000 test ones in this data set. The experiments were conducted using the TensorFlow framework. The specifications of the models are detailed in Table 1. We used stochastic gradient descent with 0.9 momentum for training FCNNs, and Adam optimization algorithm for CNNs. A suitable initial learning rate was chosen for each experiment and halved every 2,400 steps. All models were trained for 20 epochs and the results are summarized in Table 2.

Data	FCNN	CNN
MNIST	INPUT → FC 1024, ReLU → FC 1024, ReLU → FC 10	INPUT → 5x5 CONV 64, BN, ReLU, 3x3 MP → 5x5 CONV 128, BN, ReLU, 3x3 MP → FC 512, BN, ReLU → FC10
CIFAR-10	INPUT → FC 4096, ReLU → FC4096 ReLU → FC10	INPUT→ 5x5 CONV 128, BN, ReLU, 3x3 MP → 5x5 CONV 192, BN, ReLU, 3x3 MP → 5x5 CONV 256, BN, ReLU, 3x3 MP → FC 512, BN, ReLU → FC 10

Table 1: The specifications of the models used in the experiment. "INPUT" refers to the input layer (no operation is performed here). "FC d" means fully-connected layer with d hidden units. "hxh CONV c" is a convolutional layer with filter size h and c output channels. "hxh MP" is a max-pooling layer with filter size h. "BN" is a batch normalization layer. "ReLU" is the rectified linear unit.

We see that with a sufficiently large $n_l = n$ our modified models use significantly small parameters and computational cost while achieving similar results. For the fully connected neural network, our modified models can decrease the number of parameters to a third or a fourth of that of the original neural network with the top-1 error increasing less than 3%. For the convolutional neural network, our Approach II uses only about 40% and 60% number of parameters of the original CNN for MNIST and CIFAR-10 repectively, while the top-1 error increases at most 4%. All these shows the effectiveness of the random projection layer in our proposed new models. The first a few layers of a neural network is mainly for dimension reduction. In a traditional neural network architecture, weights in those layers are well trained with a relatively high cost. In our proposed model, they are replaced by a fixed random projection, which suffices for the task of dimension reduction by theories from compressed sensing. Since both MNIST and CIFAR-10 datasets consist of images, which are well known to be sparse under suitable transforms (e.g., wavelet transforms), the random projection layer can effectively map them into a low-dimensional space and we only need to train a function on this low-dimensional space. Consequently, our new neural network models with random projections can significantly reduce the number of training parameters without changing the prediction error too much, as we have seen from experimental results in Table 2.

Acknowledgments

	Data	Configuration		Top-1 Error	FLOPs (10^6)	Training Parameters (10^3)
FCNN	MNIST	Original		1.44%	3.70	1,851.39
		$n=250$		1.48%	1.93	512.00
		$n=150$		1.83%	1.16	307.20
		$n=100$		2.29%	0.77	204.80
		$n=50$		3.42%	0.39	102.40
	CIFAR-10	Original		40.69%	58.72	29,360.13
		$n=1500$		41.34%	46.08	12,288.00
		$n=1000$		41.80%	30.72	8,192.00
		$n=700$		42.89%	21.50	5,734.40
		$n=500$		43.05%	15.36	4,096.00
CNN	MNIST	Original		0.48%	82.79	206.40
		Approach I	$n=15$	0.56%	12.25	2.88
			$n=10$	0.66%	8.17	1.92
			$n=5$	0.84%	4.08	0.96
			$n=3$	1.05%	2.45	0.58
		Approach II	$n=10$	0.50%	39.78	82.56
			$n=7$	0.60%	27.85	57.79
			$n=5$	0.59%	19.89	41.28
			$n=3$	0.87%	11.93	24.77
	CIFAR-10	Original		15.15%	491.52	1,852.80
		Approach I	$n=40$	21.41%	111.98	23.04
			$n=25$	22.20%	69.99	14.40
			$n=15$	24.69%	41.99	8.64
			$n=10$	27.13%	28.00	5.76
		Approach II	$n=15$	15.62%	331.01	1,111.68
			$n=10$	17.67%	220.67	741.12
			$n=7$	18.92%	154.47	518.78
			$n=5$	20.49%	110.34	370.56

Table 2: The model errors and complexities under various configurations. "Original" means no RP scheme is applied. "FLOPs" counts the number of multiplications and additions in the model, and "Parameters" is the number of parameters. Note that "FLOPs" and "Parameters" do not take into account biases, BN, the last layer in FCNNs (which computes the predicted probabilities), or the FC layers in CNNs. We choose $n_l = n$ for all n.

References

[1] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[2] R. G. Baraniuk and M. B. Wakin. Random projections of smooth manifolds. *Found. Comput. Math.*, 9(1):51–77, 2009.

[3] A. R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine*

       learning, 14(1):115–133, 1994.

 [4] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

 [5] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *C. R. Math. Acad. Sci. Paris*, 346(9-10):589–592, 2008.

 [6] E. J. Candes, Y. C. Eldar, D. Needell, and P. Randall. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31(1):59–73, 2011.

 [7] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

 [8] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Inform. Theory*, 51(12):4203–4215, 2005.

 [9] C. K. Chui, S.-B. Lin, and D.-X. Zhou. Construction of neural networks for realization of localized deep learning. *arXiv preprint arXiv:1803.03503*, 2018.

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2(4):303–314, 1989.

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

[12] L. C. Evans and R. F. Gariepy. *Measure theory and fine properties of functions*. Textbooks in Mathematics. CRC Press, Boca Raton, FL, revised edition, 2015.

[13] S. Foucart and M.-J. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

[14] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[15] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[16] B. Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *arXiv:1708.02691*, 2017.

[17] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):356–66, 1989.

[18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[19] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pages 6232–6240, 2017.

[20] M. Lustig, D. Donoho, and J. M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic resonance in medicine*, 58(6):1182–1195, 2007.

[21] S. Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[22] E. J. McShane. Extension of range of functions. *Bull. Amer. Math. Soc.*, 40(12):837–842, 1934.

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

[24] N. Narodytska and S. Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, July 2017.

[25] J. Nash. The imbedding problem for Riemannian manifolds. *Ann. of Math. (2)*, 63:20–63, 1956.

[26] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccu-

rate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[27] X. Qu, M. Mayzel, J.-F. Cai, Z. Chen, and V. Orekhov. Accelerated nmr spectroscopy with low-rank reconstruction. *Angewandte Chemie International Edition*, 54(3):852–854, 2015.

[28] H. Rauhut, K. Schnass, and P. Vandergheynst. Compressed sensing and redundant dictionaries. *IEEE Transactions on Information Theory*, 54(5):2210–2219, 2008.

[29] M. Rudelson and R. Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

[30] W. F. Schmidt, M. A. Kraaijveld, R. P. Duin, et al. Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, pages 1–4. IEEE COMPUTER SOCIETY PRESS, 1992.

[31] U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2016.

[32] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks. *arXiv:1710. 08864*, 10 2017.

[33] H. Whitney. Analytic extensions of differentiable functions defined in closed sets. *Trans. Amer. Math. Soc.*, 36(1):63–89, 1934.