Towards Understanding the Invertibility of Convolutional Neural Networks

Anna C. Gilbert[1] **Yi Zhang**[1] **Kibok Lee**[1] **Yuting Zhang**[1] **Honglak Lee**[1,2]
[1]University of Michigan, Ann Arbor, MI 48109
[2]Google Brain, Mountain View, CA 94043
{annacg,yeezhang,kibok,yutingzh}@umich.edu, honglak@{umich.edu,google.com}

Abstract

Several recent works have empirically observed that Convolutional Neural Nets (CNNs) are (approximately) invertible. To understand this approximate invertibility phenomenon and how to leverage it more effectively, we focus on a theoretical explanation and develop a mathematical model of sparse signal recovery that is consistent with CNNs with random weights. We give an exact connection to a particular model of model-based compressive sensing (and its recovery algorithms) and random-weight CNNs. We show empirically that several learned networks are consistent with our mathematical analysis and then demonstrate that with such a simple theoretical framework, we can obtain reasonable reconstruction results on real images. We also discuss gaps between our model assumptions and the CNN trained for classification in practical scenarios.

1 Introduction

Deep learning has achieved remarkable success in many technological areas, including automatic speech recognition [Hinton *et al.*, 2012; Hannun *et al.*, 2014], natural language processing [Collobert *et al.*, 2011; Mikolov *et al.*, 2013; Cho *et al.*, 2014], and computer vision, in particular with deep Convolutional Neural Networks (CNNs) [LeCun *et al.*, 1989; Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2015; Szegedy *et al.*, 2015].

Following the unprecedented success of deep networks, there have been some theoretical works [Arora *et al.*, 2014; Arora *et al.*, 2015; Paul and Venkatasubramanian, 2014] that suggest several mathematical models for different deep learning architectures. However, theoretical analysis and understanding lag behind the very rapid evolution and empirical success of deep architectures, and more theoretical analysis is needed to better understand the state-of-the-art deep architectures, and possibly to improve them further.

In this paper, we address the gap between the empirical success and theoretical understanding of the CNNs, in particular its invertibility (i.e., reconstructing the input from the hidden activations), by analyzing a simplified mathematical model using random weights (See Section 2.1 and 4.1 for the practical relevance of the assumption).

This property is intriguing because CNNs are typically trained with discriminative objectives (i.e., unrelated to reconstruction) with a large amount of labels, such as the ImageNet dataset [Deng *et al.*, 2009]. Bruna *et al.* [2014] studied signal discovery from generalized pooling operators using image patches on non-convolutional small scale networks and datasets. Dosovitskiy and Brox [2016] used upsampling-deconvolutional architectures to invert the hidden activations of feedforward CNNs to the input domain. In another related work, Zhao *et al.* [2016] proposed a stacked what-where autoencoder network and demonstrated its promise in unsupervised and semi-supervised settings. Zhang *et al.* [2016] showed that CNNs discriminately trained for image classification (e.g., VGGNet [Simonyan and Zisserman, 2015]) are almost fully invertible using pooling switches. Despite these interesting results, there is no clear theoretical explanation as to why CNNs are invertible yet.

We introduce three new concepts that, coupled with the accepted notion that images have sparse representations, guide our understanding of CNNs:

1. we provide a *particular* model of sparse linear combinations of the learned filters that are consistent with natural images; also, this model of sparsity is itself consistent with the feedforward network;
2. we show that the effective matrices that capture explicitly the convolution of multiple filters exhibit a model-Restricted Isometry Property (model-RIP) [Baraniuk *et al.*, 2010]; and
3. our model can explain each layer of the feedforward CNN algorithm as one iteration of Iterative Hard Thresholding (IHT) [Blumensath and Davies, 2009] for model-based compressive sensing and, hence, we can reconstruct the input simply and accurately.

In other words, we give a theoretical connection to a particular version of model-based compressive sensing (and its recovery algorithms) and CNNs. Using the connection, we give a reconstruction bound for a single layer in CNNs, which can be possibly extended to multiple layers. In the experimental sections, we show empirically that large-scale CNNs are consistent with our mathematical analysis. This paper explores these properties and elucidates specific empirical aspects that further mathematical models might need to take into account.

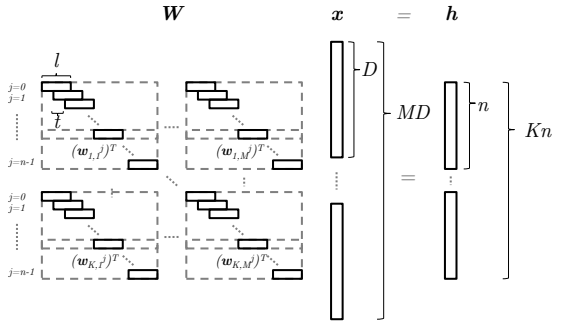


Figure 1: One-dimensional CNN architecture where $\boldsymbol{W} \in \mathbb{R}^{Kn \times MD}$ is the matrix instantiation of convolution over M channels with a filter bank consisting of K different filters. Note that a filter bank has K filters of size $l \times M$, such that there are lMK parameters in this architecture.

2 Preliminaries

In this section, we begin with discussion on the effectiveness of random weights in CNNs, and then we provide the notations for CNNs, compressive sensing, and sparse signal recovery.

2.1 Effectiveness of Gaussian Random Filters

CNNs with Gaussian random filters have been shown to be surprisingly effective in unsupervised and supervised deep learning tasks. Jarrett *et al.* [2009] showed that random filters in 2-layer CNNs work well for image classification. Also, Saxe *et al.* [2011] observed that convolutional layer followed by pooling layer is frequency selective and translation invariant, even with random filters, and these properties lead to good performance for object recognition tasks. On the other hand, Giryes *et al.* [2016] proved that CNNs with random Gaussian filters have metric preservation property, and they argued that the role of training is to select better hyperplanes discriminating classes by distorting boundary points among classes. According to their observation, random filters are in fact a good choice if training data are initially well-separated. Also, He *et al.* [2016] empirically showed that random weight CNNs can do image reconstruction well.

To better demonstrate the effectiveness of Gaussian random CNNs, we evaluate their classification performance on CIFAR-10 [Krizhevsky, 2009] in Section 4.1. Although the performance is not the state-of-the-art, it is surprisingly good considering that the networks are almost untrained. Our theoretical results may provide a new perspective on explaining these phenomena.

2.2 Convolutional Neural Nets

For simplicity, we vectorize input signals to 1-d signal; for any operations we would ordinarily carry out on images, we do on vectors with the appropriate modifications. We define a single layer of our CNN as follows. We assume that the input signal $\boldsymbol{x}$ consists of M channels, each of length D, and we write $\boldsymbol{x} \in \mathbb{R}^{MD}$. For each of the input channels, $m = 1, \ldots, M$, let $\boldsymbol{w}_{i,m}$, $i = 1, \ldots, K$ denote one of K filters, each of length ℓ. Let t be the stride length, the number of indices by which we shift each filter. Note that t can be larger than 1. We assume that the number of shifts, $n = (D - \ell)/t + 1$, is an integer. Let $\boldsymbol{w}^j_{i,m}$ be a vector of length D that consists of the (i, m)-th filter shifted by jt, $j = 0, \ldots, n-1$ (i.e., $\boldsymbol{w}^j_{i,m}$ has at most ℓ non-zero entries). We concatenate over the M channels each of these vectors (as row vectors) to form a large matrix $\boldsymbol{W}$, which is the $Kn \times MD$ matrix made up of K blocks of the n shifts of each filter in each of M channels. We assume that $Kn \geq MD$ and the Kn row vectors of $\boldsymbol{W}$ span $\mathbb{R}^{MD}$ and that we have normalized the rows so that they have unit ℓ_2 norm. The hidden units of the feed-forward CNN are computed by multiplying an input signal $\boldsymbol{x} \in \mathbb{R}^{MD}$ by the matrix $\boldsymbol{W}$ (i.e., convolving, in each channel, by a filter bank of size K, and summing over the channels to obtain Kn outputs).[1] We use $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x}$ for the hidden activation computed by a single layer CNN without pooling. Figure 1 illustrates the architecture. As a nonlinear activation, we apply the ReLU function to the Kn outputs, and then selecting the value with maximum absolute value in each of the K blocks; i.e., we perform max pooling over each of the convolved filters.

2.3 Compressive Sensing

In compressive sensing, we assume that there is a latent sparse code $\boldsymbol{z}$ that generates the visible signal $\boldsymbol{x}$. We say that a $p \times q$ matrix $\boldsymbol{\Phi}$ with $q > p$ satisfies the Restricted Isometry Property RIP(k, δ_k) if there is a distortion factor $\delta_k > 0$ such that for all $\boldsymbol{z} \in \mathbb{R}^q$ with exactly k non-zero entries, $(1 - \delta_k)\|\boldsymbol{z}\|_2^2 \leq \|\boldsymbol{\Phi}\boldsymbol{z}\|_2^2 \leq (1 + \delta_k)\|\boldsymbol{z}\|_2^2$. If $\boldsymbol{\Phi}$ satisfies RIP with sufficiently small δ_k and if $\boldsymbol{z}$ is k-sparse, then given the vector $\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{z} \in \mathbb{R}^p$, we can efficiently recover $\boldsymbol{z}$ (see Candés [2008] for more details)[2]. There are many efficient algorithms, including ℓ_1 sparse coding (e.g., ℓ_2 minimization with ℓ_1 regularization) and greedy and iterative algorithms, such as Iterative Hard Thresholding (IHT) [Blumensath and Davies, 2009].

Model-based compressive sensing. While sparse signals are a natural model for some applications, they are less realistic for CNNs. We consider a vector $\boldsymbol{z} \in \mathbb{R}^{Kn}$ as the true sparse code generating the CNN input $\boldsymbol{x}$ with a particular model of sparsity. Rather than permitting k non-zero entries anywhere in the vector $\boldsymbol{z}$, we divide the support of $\boldsymbol{z}$ into K contiguous blocks of size n and we stipulate that from each block there is at most one non-zero entry in $\boldsymbol{z}$ with a total of k non-zero entries. We call a vector with this sparsity model *model-k-sparse* and denote the union of all k-sparse subspaces with this structure $\mathcal{M}_k$. It is clear that $\mathcal{M}_k$ contains $n^k\binom{K}{k}$ subspaces. In our analysis, we consider linear combinations of two model-k-sparse signals. To be precise, suppose that $\boldsymbol{z} = \alpha_1\boldsymbol{z}_1 + \alpha_2\boldsymbol{z}_2$ is the linear combination of two elements in $\mathcal{M}_k$. Then, we say that $\boldsymbol{z}$ lies in the linear subspace $\mathcal{M}_k^2$ that consists of all linear combinations of vectors from $\mathcal{M}_k$.[3] We say that a matrix $\boldsymbol{\Phi}$ satisfies the *model-RIP* if there is a distortion factor

[1] Convolution can be computed more efficiently than matrix multiplication, but they are mathematically equivalent.

[2] We note that this is a sufficient condition and that there are other, less restrictive sufficient conditions, as well as more complicated necessary conditions.

[3] Intuitively, $\mathcal{M}_k^2$ is a subspace where the error signal $\hat{\boldsymbol{z}} - \boldsymbol{z}$ lies in and used for reconstruction bound derivation; see Appendix A.

Algorithm 1 Model-based IHT

Input: model-RIP matrix $\boldsymbol{\Phi}$, measurement $\boldsymbol{x}$ $(= \boldsymbol{\Phi}\boldsymbol{z})$, structured sparse approximation algorithm $\mathbb{M}$
Output: k-sparse approximation $\hat{\boldsymbol{z}}$
1: Initialize $\hat{\boldsymbol{z}}_0 = 0$, $\boldsymbol{d} = \boldsymbol{x}$, $i = 0$
2: **while** stopping criteria not met **do**
3: $\quad i \leftarrow i + 1$
4: $\quad \boldsymbol{b} \leftarrow \hat{\boldsymbol{z}}_{i-1} + \boldsymbol{\Phi}^T \boldsymbol{d}$
5: $\quad \hat{\boldsymbol{z}}_i \leftarrow \mathbb{M}(\boldsymbol{b}, k)$
6: $\quad \boldsymbol{d} \leftarrow \boldsymbol{x} - \boldsymbol{\Phi}\hat{\boldsymbol{z}}_i$
7: **end while**
8: **return** $\hat{\boldsymbol{z}} \leftarrow \hat{\boldsymbol{z}}_i$

$\delta_k > 0$ such that, for all $\boldsymbol{z} \in \mathcal{M}_k$,

$$(1 - \delta_k)\|\boldsymbol{z}\|_2^2 \leq \|\boldsymbol{\Phi}\boldsymbol{z}\|_2^2 \leq (1 + \delta_k)\|\boldsymbol{z}\|_2^2. \tag{1}$$

See Baraniuk *et al.* [2010] for the definitions of model sparse and model-RIP, as well as the necessary modifications to account for signal noise and compressible (as opposed to exactly sparse) signals, which we don't consider in this paper to keep our analysis simple. Intuitively, a matrix satisfying the model-RIP is a nearly orthonormal matrix of a particular set of sparse vectors with a particular sparsity model or pattern.

For our analysis, we also need matrices $\boldsymbol{\Phi}$ that satisfy the model-RIP for vectors $\boldsymbol{z} \in \mathcal{M}_k^2$. We denote the distortion factor δ_{2k} for such matrices; note that $\delta_k \leq \delta_{2k} < 1$.

Many efficient algorithms have been proposed for sparse coding and compressive sensing [Olshausen and others, 1996; Mallat and Zhang, 1993; Beck and Teboulle, 2009]. As with traditional compressive sensing, there are efficient algorithms for recovering model-k-sparse signals from measurements [Baraniuk *et al.*, 2010], assuming the existence of an efficient structured sparse approximation algorithm $\mathbb{M}$, that given an input vector and the sparsity parameter, returns the vector closest to the input with the specified sparsity structure.

In CNNs, the max pooling operator finds the downsampled activations that are closest to the activations of the original size by retaining the most significant values. The max pooling can be viewed as two steps: 1) zeroing out the locally non-maximum values; 2) downsampling the activations with the locally maximum values retained. To study the pooled activations with sparsity structures, we can recover dimension loss from the downsampling step by an unsampling operator. This procedure defines our structured sparse approximation algorithm $\hat{\boldsymbol{z}} = \mathbb{M}(\boldsymbol{h}, k)$, where $\boldsymbol{h}$ is the original (unpooled) response, k is the sparsity parameter for block-sparsification, and $\hat{\boldsymbol{z}}$ is the sparsified response after pooling but without shrinking the length (i.e., the locally non-maximum values are zeroed out such that $\hat{\boldsymbol{z}}$ has the same dimension as $\boldsymbol{h}$). Note that $\hat{\boldsymbol{z}}$ is a model-k-sparse signal by construction. On the other hand, without considering the block-sparsification, we actually apply the following max pooling and upsampling operations:

$$\hat{\boldsymbol{z}} = \text{upsample}(\text{max-pool}(\boldsymbol{h}), \boldsymbol{s}), \tag{2}$$

where $\hat{\boldsymbol{z}}$ is the pooled response, $\boldsymbol{h}$ is the filter response of CNN given input before max pooling (see Section 2.2), and $\boldsymbol{s}$ denotes the upsampling switches that indicate whereto place the non-zero values in the upsampled activations. Since our theoretical analysis does not depend on $\boldsymbol{s}$ but depends on k, any type of valid upsampling switches will be consistent with the block-sparsification (model-k-sparse) assumption, thus we will use $\mathbb{M}(\boldsymbol{h}, k)$ to denote the structured sparse approximation algorithm (2) without worrying about $\boldsymbol{s}$.

We use model-sparse version of IHT [Blumensath and Davies, 2009] as our recovery algorithm, as one iteration of IHT for our model of sparsity captures exactly a feedforward CNN.[4] Algorithm 1 describes the model-based IHT algorithm. In particular, the sequence of steps 4–6 in the middle IHT is exactly one layer of a feedforward CNN. As a result, the theoretical analysis of IHT for model-based sparse signal recovery serves as a guide for how to analyze the approximation activations of a CNN.

3 Analysis

Following the idea of compressive sensing in Section 2.3, we assume that the input $\boldsymbol{x}$ is generated from a latent model-k-sparse signal $\boldsymbol{z}$ with basis vectors $\boldsymbol{\Phi}$, which turns out to be $\boldsymbol{W}^T$ by Theorem 3.1 (i.e., $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$). Therefore, our analysis views the output of CNN (with pooling) is a reconstruction of $\boldsymbol{z}$ (i.e., $\hat{\boldsymbol{z}} = \mathbb{M}(\boldsymbol{W}\boldsymbol{x}, k)$), and $\boldsymbol{W}^T$ can be used to reconstruct $\boldsymbol{x}$ from $\hat{\boldsymbol{z}}$: that is, $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\hat{\boldsymbol{z}}$.

3.1 CNN Filters with Positive and Negative Pairs

Here we assume that all of the entries in the vectors are real numbers rather than only non-negative like when using ReLU. This setup is equivalent to using Concatenated ReLU (CReLU) [Shang *et al.*, 2016] as an activation function (i.e., keeping the positive and negative activations as separate hidden units) with tied decoding weights. The CReLU activation scheme is justified by the fact that trained CNN filters come in positive and negative pairs and that it achieves superior classification performance in several benchmarks. This setting makes a CNN much easier to analyze within the model compressed sensing framework.

To motivate the setting, we begin with a simple example. Suppose that the matrix $\boldsymbol{W}$ is an orthonormal basis for $\mathbb{R}^{MD}$ and define $\boldsymbol{\Psi} = \begin{bmatrix}\boldsymbol{W}^T & -\boldsymbol{W}^T\end{bmatrix}$.

Proposition 1. A one-layer CNN using the matrix $\boldsymbol{\Psi}^T$, with no pooling, gives perfect reconstruction (with the matrix $\boldsymbol{\Psi}$) for any input vector $\boldsymbol{x} \in \mathbb{R}^{MD}$.

Proof. Because we have both the positive and the negative dot products of the signal with the basis vectors in $\mathrm{ReLU}(\boldsymbol{\Psi}^T\boldsymbol{x}) = \mathrm{ReLU}\left(\begin{bmatrix}\boldsymbol{W}\boldsymbol{x} \\ -\boldsymbol{W}\boldsymbol{x}\end{bmatrix}\right)$, we have positive and negative versions of the hidden units $\boldsymbol{h}_+ = \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x})$ and $\boldsymbol{h}_- = \mathrm{ReLU}(-\boldsymbol{W}\boldsymbol{x})$ where we decompose $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} = \boldsymbol{h}_+ - \boldsymbol{h}_-$ into the difference of two non-negative vectors, the positive and the negative entries of $\boldsymbol{h}$. From this decomposition, we can easily reconstruct the original signal via

$$\begin{aligned}\boldsymbol{\Psi}\begin{bmatrix}\boldsymbol{h}_+ \\ \boldsymbol{h}_-\end{bmatrix} &= \begin{bmatrix}\boldsymbol{W}^T & -\boldsymbol{W}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{h}_+ \\ \boldsymbol{h}_-\end{bmatrix} = \boldsymbol{W}^T(\boldsymbol{h}_+ - \boldsymbol{h}_-) \\ &= \boldsymbol{W}^T\boldsymbol{h} = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x} = \boldsymbol{x}.\end{aligned}$$

[4] Multiple iterations of IHT can improve the quality of signal recovery. However, it is rather equivalent to the recurrent version of CNNs and does not fit to the scope of this work.

□

In the example above, we have pairs of vectors $(\boldsymbol{w}, -\boldsymbol{w})$ in our matrix $\boldsymbol{\Psi}$. Now suppose that we have a vector $\boldsymbol{z}$ where its positive and negative components can be split into $\boldsymbol{z} = \boldsymbol{z}_+ - \boldsymbol{z}_-$, and that we synthesize a signal $\boldsymbol{x}$ from $\boldsymbol{z}$ using the matrix $\begin{bmatrix}\boldsymbol{W}^T & -\boldsymbol{W}^T\end{bmatrix}$. Then, we have

$$\begin{bmatrix}\boldsymbol{W}^T & -\boldsymbol{W}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{z}_+\\ \boldsymbol{z}_-\end{bmatrix} = \boldsymbol{W}^T(\boldsymbol{z}_+ - \boldsymbol{z}_-) = \boldsymbol{W}^T\boldsymbol{z} = \boldsymbol{x}.$$

Next, we multiply $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$ by a concatenation of positive and negative $\boldsymbol{W}$, then we get $\begin{bmatrix}\boldsymbol{W}\\ -\boldsymbol{W}\end{bmatrix}\boldsymbol{x} = \begin{bmatrix}\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\\ -\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\end{bmatrix}$ and if we apply ReLU to this vector, we get $\begin{bmatrix}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z})_+\\ (\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z})_-\end{bmatrix}$, which is a vector $\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}$ that is split into its positive and negative components. The structure of the product $\boldsymbol{W}\boldsymbol{W}^T$ is crucial to the reconstruction quality of the vector $\boldsymbol{z}$. In addition, this calculation shows that if we have both positive and negative pairs of filters or vectors, then the ReLU function applied to both the positive and negative dot products simply splits the vector into the positive and negative components. These components are then reassembled in the next computation. For this reason, in the analysis in the following sections, it is sufficient to assume that all of the entries in the vectors are real numbers, rather than only non-negative.

3.2 Model-RIP and Random Filters

Our first main result shows that if we use Gaussian random filters in our CNN, then, with high probability, $\boldsymbol{W}^T$, the transpose of a large matrix formed by the convolution filters satisfies the model-RIP. In other words, Gaussian random filters generate a matrix whose transpose $\boldsymbol{W}^T$ is almost an orthonormal transform for sparse signals with a particular sparsity pattern (that is consistent with our pooling procedure). The bounds in the theorem tell us that we must balance the size of the filters ℓ and the number of channels M against the sparsity of the hidden units k, the number of the filter banks K, the number of shifts n, the distortion parameter δ_k, and the failure probability ϵ. The proof is in Appendix A.

Theorem 3.1. *Assume that we have MK vectors $\boldsymbol{w}_{i,m}$ of length ℓ in which each entry is a scaled i.i.d. (sub-)Gaussian random variable with zero mean and unit variance (the scaling factor is $1/\sqrt{M\ell}$). Let t be the stride length (where $n = (D-\ell)/t+1$) and $\boldsymbol{W}$ be a structured random matrix, which is the weight matrix of a single layer CNN with M channels and input length D. If*

$$\frac{M\ell^2}{D} \geq \frac{C}{\delta_k^2}\Big(k(\log(K)+\log(n)) - \log(\epsilon)\Big)$$

for a positive constant C, then with probability $1-\epsilon$, the $MD \times Kn$ matrix $\boldsymbol{W}^T$ satisfies the model-RIP for model $\mathcal{M}_k$ with parameter δ_k.

We also note that the same analysis can be applied to the sum of two model-k-sparse signals, with changes in the constants (that we do not track here).

Corollary 3.2. *Random matrices with the CNN structure satisfy, with high probability, the model-RIP for $\mathcal{M}_k^2$.*

Other examples of matrices that satisfy the model-RIP include wavelets and localized Fourier bases; both examples can be easily and efficiently implemented via convolutions.

3.3 Reconstruction Bounds

Suppose $\boldsymbol{W}^T$ satisfies the model-RIP and $\hat{\boldsymbol{z}}$ is the reconstruction of true sparse code $\boldsymbol{z}$ through a CNN layer followed by pooling, i.e., $\hat{\boldsymbol{z}} = \mathbb{M}(\boldsymbol{W}\boldsymbol{x}, k)$. Then, Theorem 3.3 shows that $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\hat{\boldsymbol{z}}$ is an approximate reconstruction of the input signal, and the relative error is bounded on a function of the distortion parameters of the model-RIP.

Theorem 3.3. *We assume that $\boldsymbol{W}^T$ satisfies the $\mathcal{M}_k^2$-RIP with constant $\delta_k \leq \delta_{2k} < 1$. If we use $\boldsymbol{W}$ in a single layer CNN both to compute the hidden units $\hat{\boldsymbol{z}}$ and to reconstruct the input $\boldsymbol{x}$ from these hidden units as $\hat{\boldsymbol{x}}$ so that $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\mathbb{M}(\boldsymbol{W}\boldsymbol{x}, k)$, the error in our reconstruction is*

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2 \leq \frac{5\delta_{2k}}{1-\delta_k}\frac{\sqrt{1+\delta_{2k}}}{\sqrt{1-\delta_{2k}}}\|\boldsymbol{x}\|_2.$$

See Appendix B for the detailed proofs. Part of our analysis also shows that the hidden units $\hat{\boldsymbol{z}}$ are approximately the putative coefficient vector $\boldsymbol{z}$ in the sparse linear representation for the input signal. Recall that the structured sparsity approximation algorithm $\mathbb{M}$ includes the downsampling caused by pooling and an unsampling operator. Theorem 3.3 is applicable to any type of upsampling switches, so our reconstruction bound is generic to the particular design choice on how to recover the activation size in a decoding neural network. We can extend the analysis for a single layer CNN to multiple layer CNN by using the output on one layer as the input to another, following the proof in Appendix B. We leave further investigation of this idea as future work.

4 Experimental Evidence and Analysis

In this section, we provide experimental validation of our theoretical model and analysis. We first validate the practical relevance of our assumption by examining the effectiveness of random filter CNNs, and then provide results on more realistic scenarios. In particular, we study popular deep CNNs trained for image classification on ILSVRC 2012 dataset [Deng *et al.*, 2009]. We calculate empirical model-RIP bounds for $\boldsymbol{W}^T$, showing that they are consistent with our theory. Our results are also consistent with a long line of research shows that it is reasonable to model real and natural images as sparse linear combinations overcomplete dictionaries [Boureau *et al.*, 2008; Le *et al.*, 2013; Lee *et al.*, 2008; Olshausen and others, 1996; Ranzato *et al.*, 2007; Yang *et al.*, 2010]. In addition, we verify our theoretical bounds for the reconstruction error $\|\boldsymbol{x} - \boldsymbol{W}^T\hat{\boldsymbol{z}}\|_2/\|\boldsymbol{x}\|_2$ on real images. We investigate both randomly sampled filters and empirically learned filters in these experiments. Our implementation is based on Caffe [Jia *et al.*, 2014] and MatConvNet [Vedaldi and Lenc, 2015].

Recall that our theoretical analysis is generic to any upsampling switches in (2) for reconstruction. In the experiments, we specifically use the naive upsampling to reverse max-pool activations to its original size, where only the first element in a pooling region is assigned with the pooled activation, and the rest elements are all zero. Thus, no extra information other

Method	1 layer	2 layers	3 layers
Random filters	66.5%	74.6%	74.8%
Learned filters	68.1%	83.3%	89.3%

Table 1: Classification accuracy of CNNs with random and learnable filters on CIFAR-10. A typical layer consists of four operators: convolution, ReLU, batch normalization and max pooling. Networks with optimal filter size and numbers of output channels are used. (See Appendix C.1 for more details about the architectures). The random filters, assumed in our theoretical analysis, perform reasonably well, not far off the learned filters.

than the pooled activation values are taken into account.

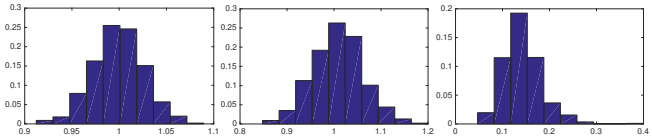
4.1 Gaussian Random CNNs on CIFAR-10

To show the practical relevance of our theoretical assumptions on using random filters for CNNs as stated in Section 2.1, we evaluate simple CNNs with Gaussian random filters with i.i.d. zero mean unit variance entries on the CIFAR-10 [Krizhevsky, 2009]. Note that the goal of this experiment is not to achieve state-of-the-art results, but to examine practical relevance of our assumption on random filter CNNs. Once the CNNs weights are initialized (randomly), they are fixed during the training of the classifiers.[5] Specifically, we test random CNNs with 1, 2, and 3 convolutional layers followed by ReLU activation and 2×2 max pooling layer. We tested different filter sizes $(3, 5, 7)$ and numbers of channels $(64, 128, 256, 1024, 2048)$ and report the best classification accuracy by cross-validation in Table 1. We also report the best performance using learnable filters for comparison. More details about the architectures can be found in Appendix C.1. We observe that CNNs with Gaussian random filters achieve good classification performance (implying that they serve as reasonable representation of input data), which is not too far off the learned filters. Our experimental results are also consistent with the observations made by Jarrett *et al.* [2009] and Saxe *et al.* [2011]. In conclusion, those results suggest that CNNs with Gaussian random filters might be a reasonable setup which is amenable to mathematical analysis while not being too far off in terms of practical relevance.

4.2 1-d Model-RIP

We use 1-d synthetic data to empirically show the basic validity of our theory in terms of the model-RIP in (1) and reconstruction bound in Theorem 3.3. We plot the histograms of the empirical model-RIP values of 1-d Gaussian random filters $\boldsymbol{W}$ (scaled by $1/\sqrt{lM}$) with size $l\times1\times M\times K = 5\times1\times32\times96$ on 1-d $\mathcal{M}_k$ sparse signal $\boldsymbol{z}$ with size $D = 32$ and sparsity $k = 10$, whose non-zero elements are drawn from a uniform distribution on $[-1, 1]$. The histograms in Figure 2 (a)–(b) are tightly centered around 1, suggesting that $\boldsymbol{W}^T$ satisfies the model-RIP in (1) and its corollary from Lemma B.1, respectively. We also empirically show the reconstruction bound in Theorem 3.3 on synthetic vectors $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$ (Figure 2 (c)). The reconstruction error is concentrated at around 0.1–0.2 and bound under 0.5. Results in Figure 2 suggest the practical validity of our theory when the model assumptions hold.

[5]Implementation detail: we add a batch normalization layer together with a learnable scale and bias before the activation so that we do not need to tune the scale of the filters. See Appendix C.1 for more details.



(a) $\|\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (b) $\|\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (c) $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2/\|\boldsymbol{x}\|_2$

Figure 2: For 1-d scaled Gaussian random filters $\boldsymbol{W}$, we plot the histogram of ratios (a) $\|\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (model-RIP in (1); supposed to be concentrated at 1), (b) $\|\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (ratio between the norm of the reconstructed code $\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}$ and that of the original code $\boldsymbol{z}$; supposed to be concentrated at 1), and (c) $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2/\|\boldsymbol{x}\|_2$ (reconstruction bound in Theorem 3.3, supposed to be small), where $\boldsymbol{z}$ is a $\mathcal{M}_k$ sparse signal that generates the vector $\boldsymbol{x}$ and $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\mathbb{M}(\boldsymbol{W}\boldsymbol{x}, k)$ is the reconstruction of $\boldsymbol{x}$, where we use the naive unsampling to recover the reduced dimension due to pooling: we place recovered values in the top-left corner in each unsampled block. (See Section 2.3).

4.3 Architectures for 2-d Model-RIP

We conduct the rest of our experimental evaluations on the 16-layer VGGNet (Model D in Simonyan and Zisserman [2015]), where the computation is carried out on images; e.g., convolution with a 2-d filter bank and pooling on square regions. In contrast to the theory, the realistic network does not pool activations over all the possible shifts for each filter, but rather on non-overlapping patches. The networks are trained for the large-scale image classification task, which is important for extending to other supervised tasks in vision. The main findings on VGGNet are presented in the rest of this section; we also provide some analysis on AlexNet [Krizhevsky *et al.*, 2012] in Appendix C.2.

VGGNet contains five macro layers of convolution and pooling layers, and each macro layer has 2 or 3 convolutional layers followed by a pooling layer. We denote the j-th convolutional layer in the i-th macro layer "conv(i, j)," and the pooling layer "pool(i)." The activations/features from i-th macro layer are the output of pool(i). Our analysis is for single convolutional layers.

4.4 2-d Model-RIP

The key to our reconstruction bound is Theorem 3.3. We empirically evaluate the model-RIP, i.e., $\|\boldsymbol{W}^T\boldsymbol{z}\|/\|\boldsymbol{z}\|$, for real CNN filters of the pretrained VGGNet. We use two-dimensional coefficients $\boldsymbol{z}$ (each block of coefficients is of size $D\times D$), K filters of size $\ell\times\ell$, and pool the coefficients over smaller pooling regions (i.e., not over all possible shifts of each filter). The following experimental evidence suggests that the sparsity model and the model-RIP of the filters are consistent with our mathematical analysis on the simpler one-dimensional case.

To check the significance of the model-RIP (i.e., how close $\|\boldsymbol{W}^T\boldsymbol{z}\|/\|\boldsymbol{z}\|$ is to 1) in controlled settings, we first synthesize the hidden activations $\boldsymbol{z}$ with sparse uniform random variables, which fully agree with our model assumptions.

The sparsity of $\boldsymbol{z}$ is constrained to the average level of the real CNN activations, which is reported in Table 2. Given the

layer	c(1,1)	c(1,2)	p(1)	c(2,1)	c(2,2)	p(2)
% of non-zeros	49.1	69.7	80.8	67.4	49.7	70.7
layer	c(3,1)	c(3,2)	c(3,3)	p(3)	c(4,1)	c(4,2)
% of non-zeros	53.4	51.9	28.7	45.9	35.6	29.6
layer	c(4,3)	p(4)	c(5,1)	c(5,2)	c(5,3)	p(5)
% of non-zeros	12.6	23.1	23.9	20.6	7.3	13.1

Table 2: Layer-wise sparsity of VGGNet on ILSVRC 2012 validation set. "c" stands for convolutional layers and "p" represents pooling layers. CNN with random filters in Section 4.4 can be simulated with the same sparsity.

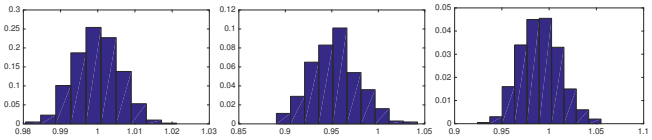
layer	(1,1)	(1,2)	(2,1)	(2,2)	(3,1)	(3,2)	(3,3)
learned	0.943	0.734	0.644	0.747	0.584	0.484	0.519
random	0.670	0.122	0.155	0.105	0.110	0.090	0.080
layer	(4,1)	(4,2)	(4,3)	(5,1)	(5,2)	(5,3)	
learned	0.460	0.457	0.404	0.410	0.410	0.405	
random	0.092	0.062	0.062	0.070	0.067	0.067	

Table 3: Comparison of coherence between learned filters in each convolutional layer of VGGNet and Gaussian random filters with corresponding sizes.

filters of a certain convolutional layer, we use the synthetic $\boldsymbol{z}$ (in equal position to this layer's output activations) to get statistics for the model-RIP. To be consistent with succeeding experiments, we choose conv$(5, 2)$, while other layers show similar results. Figure 3 (a) summarizes the distribution of empirical model-RIP values, which is clearly centered around 1 and satisfies (1) with a short tail roughly bounded by $\delta_k < 1$. For more details of the algorithm, we normalize the filters from the conv$(5, 2)$ layer, which are $\ell \times \ell$ ($\ell = 3$). All $K = 512$ filters with $M = 512$ input channels are used.[6] We set $D = 15$, which is the same as the output activations of conv$(5, 2)$, and use 2×2 pooling regions[7], which is commonly used in recent CNNs. We generate 1000 $\mathcal{M}_k$ randomly sampled sparse activation maps $\boldsymbol{z}$ by first sampling their non-zero supports and then filling elements on the supports uniformly from $[-1, 1]$. The sparsity is the same as that in conv$(5, 1)$ activations.

More realistically, we observe that the actual conv$(5, 2)$ activations from VGGNet are not necessarily drawn from a model-sparse uniform distribution. This motivates us to evaluate the empirical model-RIP on the hidden activations $\boldsymbol{z}$ that reconstruct the actual input activations $\boldsymbol{x}$ from conv$(5, 1)$ by $\boldsymbol{W}^T\boldsymbol{z}$. Per theory, the $\boldsymbol{x}$ is given by a max pooling layer, so we constrain the sparsity (i.e., the size of the support set is no more than 1 in a pooling region for a single channel). We use a simple and efficient algorithm to recover $\boldsymbol{z}$ from $\boldsymbol{x}$ in Algorithm 2. The algorithm is inspired by "ℓ_1 heuristic" method that are commonly used in practice (e.g., Boyd [2015]). As shown in Algorithm 2, we first do ℓ_1-regularized least squares without constraining the support set. Max pooling is then applied to figure out the support set for each pooling region. In particular, we use max pooling and unpooling with known switches (line 2) to zero out the locally non-maximum values without messing up the support structures. We perform ℓ_1-regularized least squares again on the fixed support set to recover the hidden activations satisfying the model sparsity. As shown in Figures 3 (b)–(c), the empirical model-RIP values for visual activations $\boldsymbol{x}$ from conv$(5, 1)$ with/without ReLU are both close to 1. The center offset to 1 is less than 0.05 and the range bound δ_k is roughly less than 0.05, which agrees with the theoretical bound in (1).

To gain more insight, we summarize the learned filter coherence in Table 3 for all convolutional layers in VGGNet.[8] This measures the correlation or similarity between the columns of W^T and is a proxy for the value of the model-RIP parameter δ_k (which we can only estimate computationally). The smaller the coherence, the smaller δ_k is, and the better the reconstruction. The coherence of the learned filters is not low, which is inconsistent with our theoretical assumptions. However, the model-RIP turns out to be robust to this mismatch. It demonstrates the strong practical invertibility of CNN.



(a) Random (b) After ReLU (c) Before ReLU

Figure 3: For VGGNet's conv$(5, 2)$ filters $\boldsymbol{W}$, we plot the histogram of ratios $\|\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$, which is expected to be concentrated at 1 according to (1), where $\boldsymbol{z}$ is a $\mathcal{M}_k$ sparse signal. In (a), $\boldsymbol{z}$ is randomly generated with the same sparsity as the conv$(5, 2)$ activations and from a uniform distribution for the non-zero magnitude. In (b) and (c), $\boldsymbol{z}$ is recovered by Algorithm 2 from the conv(5,1) activations before and after applying ReLU, respectively. The learned filters admits similar model-RIP value distributions to the random filters except for a bit larger bandwidth, which means the model-RIP in (1) can empirically hold even when the filters do not necessarily subject to the i.i.d Gaussian random assumption.

Algorithm 2 Sparse hidden activation recovery

Input: convolution matrix $\boldsymbol{W}$, input activation/image $\boldsymbol{x}$
Output: hidden code $\boldsymbol{z}$, satisfying our model-RIP assumption with $\mathcal{M}_k$ and reconstructing $\boldsymbol{x}$ with $\boldsymbol{W}$

1: $\boldsymbol{z}^{\text{init}} = \text{argmin}_{\boldsymbol{z}} \|\boldsymbol{x} - \boldsymbol{W}^T\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{z}\|_1$
2: $\boldsymbol{z}^{\text{model}} = \text{up-sample}(\text{max-pool}(\boldsymbol{z}^{\text{init}}), \boldsymbol{s})$, where $\boldsymbol{s} = \text{pool-switch}(\boldsymbol{z})$
3: $\boldsymbol{z} = \text{argmin}_{\boldsymbol{z}} \|\boldsymbol{x} - \boldsymbol{W}^T\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{z}\|_1$, s.t. $\boldsymbol{z}_i = 0$ if $\boldsymbol{z}_i^{\text{model}} = 0$

4.5 Reconstruction Bounds

With model-RIP as a sufficient condition, Theorem 3.3 provides a theoretical bound for layer-wise reconstruction via $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\mathbb{M}(\boldsymbol{W}\boldsymbol{x})$, which consists of the projection and reconstruction in one IHT iteration. Without confusion, we refer to it as IHT for notational convenience. We investigate the practical reconstruction errors on pool(1) to (4) of VGGNet.

To encode and reconstruct intermediate activations of CNNs, we employ IHT with sparsity estimated from the real CNN activations on ILSVRC 2012 validation set (see Table 2). We also reconstruct input images, since CNN inversion is not

[6] We do not remove any filters including those in approximate positive/negative pairs (see Section 3.)

[7] No pooling layer follows conv$(5, 2)$ in VGGNet. However, we use it in this way to analyze the convolution-pooling pair per theory.

[8] The coherence is defined as the maximum (in absolute value) dot product between distinct pairs of columns of the matrix W^T, i.e. $\mu = \max_{i \neq j} |W_i W_j^T|$, where W_i denote the i-th row of matrix W.

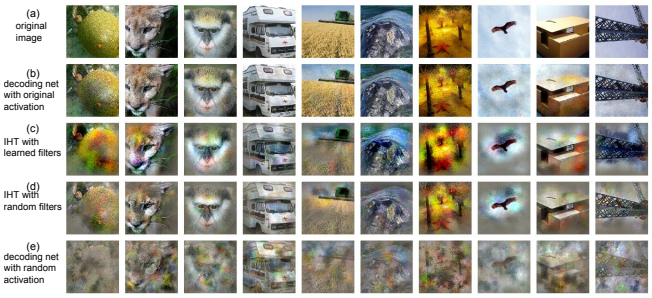


Figure 4: Visualization of images reconstructed by a pretrained decoding network with VGGNet's pool(4) activation reconstructed using different methods: (a) original image, (b) output of the 5-layer decoding network with original activation, (c) output of the decoding net with reconstructed activation by IHT with learned filters, (d) output of the decoding net with reconstructed activation by IHT with Gaussian random filters, and (e) output of the decoding net with Gaussian random activation.

limited to a single layer, and images are easier to visualize than hidden activations. To implement image reconstruction, we project the reconstructed activations into the image space via a pretrained decoding network as in Zhang *et al.* [2016], which extends a similar autoencoder architecture as in Dosovitskiy and Brox [2016] to a stacked "what-where" autoencoder [Zhao *et al.*, 2016]. The reconstructed activations were scaled to have the same norm as the original activations so that we can feed them into the decoding network.

As an example, Figure 4 illustrates the image reconstruction results for the hidden activations of pool(4). Interestingly, the decoding network itself is quite powerful, since it can reconstruct the rough (although very noisy) glimpse of images with Gaussian random input, as shown in Figure 4 (e). Object shapes are recovered up to some extent by using the pooling switches only in the "what-where" autoencoder. This result suggests that it is important to determine which pooling units are active and then to estimate these values accurately. These steps are consistent with the steps in the inner loop of any iterative sparse signal reconstruction algorithm.

In Figure 4 (c), we take the pretrained conv$(5,1)$ filters for IHT. The images recovered from the IHT reconstructed pool(4) activations are reasonable and the reconstruction quality is significantly better than the random input baseline. We also try Gaussian random filters (Figure 4 (d)), which agree more with the model assumptions (e.g., lower coherence, see Table 3). The learned filters from VGGNet perform equally well visually. IHT ties the encoder and decoder weights (no filter learning for the decoder), so it does not perform as well as the decoding network trained with a huge batch of data (Figure 4 (b)). Nevertheless, we show both theoretically and experimentally decent reconstruction bounds for these simple reconstruction methods on real CNNs. More visualization results for more layers are in Appendix C.3.

In Table 4, we summarize reconstruction performance for all 4 macro layers. With random filters, the model assumptions hold and the IHT reconstruction is the best quantitatively. IHT with real CNN filters performs comparable to the best case and much better than the baseline established by the randomly sampled activations.

layer	image space relative error			activation space relative error		
	learned filters	random filters	random activations	learned filters	random filters	random activations
1	0.423	0.380	0.610	0.895	0.872	1.414
2	0.692	0.438	0.864	0.961	0.926	1.414
3	0.326	0.345	0.652	0.912	0.862	1.414
4	0.379	0.357	0.436	1.051	0.992	1.414

Table 4: Layer-wise relative reconstruction errors by different methods in activation space and image space between reconstructed and original activations. For macro layer i, we take its activation after pooling from that layer and reconstruct it with different methods (using learned filters from the layer above or scaled Gaussian random filters) and feed the reconstructed activation to a pretrained corresponding decoding network.[9]

5 Conclusion

We introduce three concepts that tie together a particular model of compressive sensing (and the associated recovery algorithms), the properties of learned filters, and the empirical observation that CNNs are (approximately) invertible. Our experiments show that filters in trained CNNs are consistent with the mathematical properties we present while the hidden units exhibit a much richer structure than mathematical analysis suggests. Perhaps simply moving towards a compressive, rather than exactly sparse, model for the hidden units will capture the sophisticated structure in these layers of a CNN or, perhaps, we need a more sophisticated model. Our experiments also demonstrate that there is considerable information captured in the switch units (or the identities of the non-zeros in the hidden units after pooling) that no mathematical model has yet expressed or explored thoroughly. We leave such explorations as future work.

Acknowledgments

This work was supported in part by ONR N00014-16-1-2928, NSF CAREER IIS-1453651, and Sloan Research Fellowship. We would like to thank Michael Wakin for helpful discussions about concentration of measure for structured random matrices.

Full version: https://arxiv.org/abs/1705.08664

References

[Arora *et al.*, 2014] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable Bounds for Learning Some Deep Representations. In *ICML*, 2014.

[Arora *et al.*, 2015] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. Why are deep nets reversible: A simple theory, with implications for training. *arXiv:1511.05653*, 2015.

[Baraniuk *et al.*, 2010] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-Based Compressive Sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Science*, 2:183–202, 2009.

[9] The values in the last column are identical ($= 1.414$) for all layers because $\|\boldsymbol{f} - \hat{\boldsymbol{f}}\|/\|\boldsymbol{f}\| = \sqrt{2}$ on average for Gaussian random $\hat{\boldsymbol{f}}$ provided $\|\boldsymbol{f}\| = \|\hat{\boldsymbol{f}}\|$.

[Blumensath and Davies, 2009] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[Boureau *et al.*, 2008] Y-lan Boureau, Yann L Cun, et al. Sparse feature learning for deep belief networks. In *NIPS*, 2008.

[Boyd, 2015] Stephen Boyd. $l1$-norm methods for convex-cardinality problems, ee364b: Convex optimization II lecture notes, 2014-2015 spring. 2015.

[Bruna *et al.*, 2014] Joan Bruna, Arthur Szlam, and Yann LeCun. Signal recovery from pooling representations. In *ICML*, 2014.

[Candés, 2008] Emmanuel J. Candés. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9):589–592, 2008.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[Dosovitskiy and Brox, 2016] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016.

[Giryes *et al.*, 2016] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[Hannun *et al.*, 2014] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv:1412.5567*, 2014.

[He *et al.*, 2016] Kun He, Yan Wang, and John Hopcroft. A powerful generative model using random weights for the deep image representation. *arXiv:1606.04801*, 2016.

[Hinton *et al.*, 2012] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[Jarrett *et al.*, 2009] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[Krizhevsky *et al.*, 2012] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[Le *et al.*, 2013] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2013.

[LeCun *et al.*, 1989] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[Lee *et al.*, 2008] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *NIPS*, 2008.

[Mallat and Zhang, 1993] Stephane Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397 – 3415, 1993.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Olshausen and others, 1996] Bruno A Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[Park *et al.*, 2011] Jae Young Park, Han Lun Yap, C.J. Rozell, and M. B. Wakin. Concentration of Measure for Block Diagonal Matrices With Applications to Compressive Signal Processing. *IEEE Transactions on Signal Processing*, 59(12):5859–5875, 2011.

[Paul and Venkatasubramanian, 2014] Arnab Paul and Suresh Venkatasubramanian. Why does Deep Learning work? - A perspective from Group Theory. *arXiv.org*, December 2014.

[Ranzato *et al.*, 2007] Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

[Saxe *et al.*, 2011] Andrew Saxe, Pang W Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *ICML*, 2011.

[Shang *et al.*, 2016] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016.

[Simonyan and Zisserman, 2015] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[Vedaldi and Lenc, 2015] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM International Conference on Multimedia*, 2015.

[Vershynin, 2010] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv:1011.3027*, November 2010.

[Yang *et al.*, 2010] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[Zhang *et al.*, 2016] Yuting Zhang, Kibok Lee, and Honglak Lee. Augmenting neural networks with reconstructive decoding pathways for large-scale image classification. In *ICML*, 2016.

[Zhao *et al.*, 2016] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv:1506.02351*, 2016.