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,honglak}@umich.edu

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 (Bengio et al., 2013; Schmidhuber, 2015), including computer vision (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan and Zisserman, 2015), 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), bioinformatics (Chicco et al., 2014), even high energy particle physics (Baldi et al., 2014). In particular, deep Convolutional Neural Networks (CNNs) (LeCun et al., 1989; Krizhevsky et al., 2012; Simonyan and Zisserman, 2015) have been a critical enabling technique for analyzing images and sequential data.

Following the unprecedented success of deep networks, there has been some theoretical work (e.g., Arora et al. (2014; 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 attempt to address the gap between the empirical success and theoretical understanding of the Convolutional Neural Nets, in particular its invertibility (i.e., reconstructing the input from the hidden activations), by analyzing a simplified mathematical model using random weights.[1]

This property is intriguing because convolutional neural networks are typically trained with discriminative objectives (i.e., unrelated to reconstruction) with a large amount of labels, such as the ImageNet dataset. For example, Dosovitskiy and Brox (2016) used upsampling-deconvolutional architectures to invert the hidden activations of feedforward CNNs to the input domain. In other related work, Zhao et al. (2016) proposed stacked a what-where network via a (deconvolutional) decoder and demonstrate its promise in unsupervised and semi-supervised settings. Bruna et al. (2014) studied signal discovery from generalized pooling operators using image patches on non-convolutional small scale networks and datasets. Zhang et al. (2016) showed that CNNs discriminately trained for image classification (e.g., VGG Net (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]For more discussion about random filters, see Sections 2.1 and 4.1.

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 model of model-based compressive sensing (and its recovery algorithms) and CNNs. We show empirically that large-scale deep convolution networks are consistent with our mathematical analysis. We then demonstrate that with such a simple theoretical framework, we can obtain reasonable reconstruction results on real images, using filters from trained networks. Finally, we observe that it makes a significant difference which filters one uses for encoding and decoding, whether they are trained specifically for reconstruction, or random, or the same for both procedures. This paper explores these properties and elucidate specific empirical aspects that any more sophisticated mathematical model should take into account.[2]

2 Preliminaries

In this section, we set the stage for our mathematical analysis in Section 3. We begin with discussion on the use of random weights in (convolutional) neural networks, and then provide the definitions and models for CNNs. Then, we discuss compressive sensing and sparse signal recovery. We define a particular model of sparsity that we will use throughout our analysis and detail the Iterative Hard Thresholding (IHT) algorithm which is the basis of our reconstruction analysis.

In order to simplify our notation and to make clear our analysis, we focus on a single layer in the analysis instead of multiple layers.[3] Also, we assume that all of our input signals are vectors rather than matrices and that any operations we would ordinarily carry out on images (e.g., convolving with a filter bank, dividing into regions over which we pool coefficients), we do on vectors with the appropriate modifications for a simplified structure. While these assumptions ease our exposition, they do not change the nature of our arguments nor their implications for images. Furthermore, we demonstrate the validity of our results in two-dimensional natural images.

2.1 Effectiveness of Gaussian Random Filters

We analyze theoretically CNNs with Gaussian random filters, which have been 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. In addition, 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; see Section 4.1 for details. We find that a 3-layer Gaussian random CNN is able to achieve $\sim 75\%$ accuracy on the test set, with only the last classifier layer optimized, (see Table 1 for more details). Even though this number is far from the state-of-the-art results, it is surprisingly good considering the networks are almost untrained. Our theoretical results may provide another new perspective on explaining these phenomena.

[2]We note that our model may not be an exact replica of a real setting, but for mathematical analysis, it is a simplified but representative abstraction of practical settings. A number of works show that random weight CNNs still achieve surprisingly good classification accuracy although they may not match the state-of-the-art results; see Sections 2.1 and 4.1 for more discussion.

[3]We can extend the equivalency on a single layer of CNNs to multiple layer CNNs simply by using the output on one layer as the input to another, still using the steps of the inner loop of IHT.

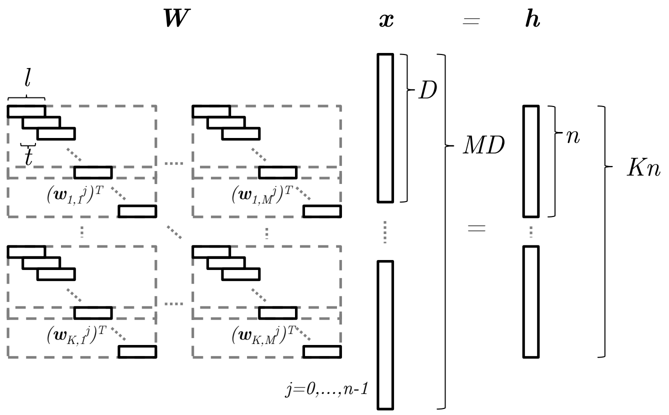


Figure 1: One-dimensional CNN architecture where $\boldsymbol{W} \in \mathbb{R}^{Kn \times MD}$ is the matrix instantiation of convolution over M channels with a filter bank consisting of K different filters. Note that a filter bank has K filters of size $l \times M$, such that there are lMK parameters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

2.2 Convolutional Neural Nets

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, $m = 1, \dots, M$, let $\boldsymbol{w}_{i,m}$, $i = 1, \dots, K$ 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 $\boldsymbol{w}_{i,m}^{j}$ be a vector of length D that consists of the (i, m)-th filter shifted by jt, $j = 0, \dots, n-1$ (i.e., $\boldsymbol{w}_{i,m}^{j}$ has at most ℓ non-zero entries). We will 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 $Kn \geq MD$. We also assume that 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. We assume that 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), applying 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 and sum over the channels.[4] 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.

2.3 Compressive sensing

Let $\boldsymbol{\Phi}$ be a $i \times j$ matrix with $j > i$. We say that $\boldsymbol{\Phi}$ satisfies the Restricted Isometry Property RIP(k, δ_k) (or, just RIP) if there is a distortion factor $\delta_k > 0$ such that for all $\boldsymbol{z} \in \mathbb{R}^j$ with exactly k non-zero entries, $(1 - \delta_k)\|\boldsymbol{z}\|_2^2 \leq \|\boldsymbol{\Phi z}\|_2^2 \leq (1 + \delta_k)\|\boldsymbol{z}\|_2^2$. If $\boldsymbol{\Phi}$ satisfies RIP (for appropriate sparsity level k and sufficiently small δ_k) and if $\boldsymbol{z} \in \mathbb{R}^j$ is k-sparse, then, given the vector $\boldsymbol{x} = \boldsymbol{\Phi z} \in \mathbb{R}^i$, we can efficiently recover $\boldsymbol{z}$ (see Candés (2008) for more details)[5]. There are many efficient algorithms for doing so, including ℓ_1 sparse coding (e.g., ℓ_2 minimization with ℓ_1 regularization) and greedy, iterative algorithms (such as Iterative Hard Thresholding or IHT).

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 for 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

[4]The convolution can be computed more efficiently than a straight-forward matrix multiplication, but they are mathematically equivalent.

[5]We note that this is a sufficient condition and that there are other, less restrictive sufficient conditions, as well as more complicated necessary conditions. Furthermore, we have not given the exact, quantitative relations amongst the parameters. For simplicity, we stick with this definition.

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 $\mathcal{M}_k$.

We say that a matrix $\boldsymbol{\Phi}$ satisfies the **model-RIP condition** for parameter k if, there is a distortion factor $\delta_k > 0$ such that, for all $\boldsymbol{z} \in \mathcal{M}_k$,

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

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 have neglected to consider to keep our analysis simple). Intuitively speaking, a matrix that satisfies the model-RIP is a nearly an orthonormal matrix for 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 condition for vectors $\boldsymbol{z} \in \mathcal{M}_k^2$. We denote the distortion factor δ_{2k} for such matrices. Note that $\delta_k \leq \delta_{2k} < 1$.

Algorithm 1 Model-based IHT

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

Many efficient algorithms have been proposed for sparse coding and compressive sensing (Olshausen et al., 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 (see 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 convolutional neural networks, 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 second step (downsampling step) by an unsampling operator. This procedure defines our structured sparse approximation algorithm $\mathbb{M}(\boldsymbol{z}, k)$, where $\boldsymbol{z}$ is the original (unpooled) code, and k is the sparsity parameter for further sparsification, which guarantees that $\mathbb{M}(\boldsymbol{z}, k)$ is a model-k-sparse signal. With the standard layered formulation for neural networks, we have

$$\mathbb{M}(\boldsymbol{z}, k) = \text{block-sparsify}(\text{upsample}(\text{max-pool}(\boldsymbol{z}), \boldsymbol{s}), k), \tag{2}$$

where $\boldsymbol{s}$ denotes the upsampling switches that indicate where to place the non-zero values in the upsampled activations. Taking the pooling switches known from the max pooling operation as $\boldsymbol{s}$, we specifically define $\mathbb{M}$ as the nesting of the max pooling and the unpooling with known switch. We define this special case as

$$\mathbb{M}_{\text{known}}(\boldsymbol{z}, k) = \text{block-sparsify}(\text{upsample}(\text{max-pool}(\boldsymbol{z}), \text{max-pool-switch}(\boldsymbol{z})), k). \tag{3}$$

Alternatively, using the fixed uniform switches as $\boldsymbol{s}$, we specifically define $\mathbb{M}$ as the nesting of the max pooling and the naive unsampling, denoted by $\mathbb{M}_{\text{fixed}}$. In the rest of this paper, our theoretical analysis are generic to any type of valid upsampling switches[6], so we use $\mathbb{M}(\boldsymbol{z}, k)$ to denote the structured sparse approximation algorithm without worrying about $\boldsymbol{s}$. The two special cases $\mathbb{M}_{\text{known}}$ and $\mathbb{M}_{\text{fixed}}$ are used in the empirical analysis when we need to specify $\mathbb{M}(\boldsymbol{z}, k)$ as a fully concrete operator.

The main recovery algorithm that we focus on is a model-sparse version of Iterative Hard Thresholding (IHT) (see Blumensath and Davies (2009)), not because we are interested in recovering model-sparse signals, per se, but because one iteration of IHT for our model of sparsity captures exactly

[6]Valid switches should place a non-zero value at exactly one location.

a feedforward CNN.[7] Algorithm 1 describes the model-based IHT algorithm. In particular, the sequence of steps 4–6 in the middle IHT (without the outer iterative loop) 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

To motivate our more formal analysis, 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

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

□

In the example above, we have pairs of vectors $(\boldsymbol{w}, -\boldsymbol{w})$ in our matrix $\boldsymbol{\Psi}$. This settings allow us to turn what would ordinarily be a nonlinear function, ReLU, into a linear one. In fact, the assumption that trained CNN filters come in positive and negative is validated by Shang et al. (2016), which makes a CNN much easier to analyze within the model compressed sensing framework.

Suppose that we have a vector $\boldsymbol{z}$ that we split into positive and negative components, $\boldsymbol{z} = \boldsymbol{z}_+ - \boldsymbol{z}_-$, and that we synthesize (or construct) 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, suppose that we multiply $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$ by the transpose of the same matrix, we find $\begin{bmatrix}\boldsymbol{W}\\ -\boldsymbol{W}\end{bmatrix}\boldsymbol{x} = \begin{bmatrix}\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\\ -\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\end{bmatrix}$ and, if we apply ReLU to this vector, we produce $\begin{bmatrix}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z})_+\\ (\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z})_-\end{bmatrix}$ a vector that is split into its positive and negative components. To determine whether or not we have "reconstructed" the vector $\boldsymbol{z}$, the structure of the product $\boldsymbol{W}\boldsymbol{W}^T$ is crucial. 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 consider $\boldsymbol{W}^T\boldsymbol{z} = \boldsymbol{x}$ and $\boldsymbol{W}\boldsymbol{x} = \boldsymbol{h}$ with max pooling alone applied to $\boldsymbol{h}$, assuming that all of the entries in the vectors are real numbers, rather than only non-negative.

3.1 Model-RIP and random filters

Our first main result says that if we use Gaussian random filters in our CNN, then, with high probability, the transpose of the matrix $\boldsymbol{W}$ formed by the convolutions with these filters has the model-RIP property. 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.

[7]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.

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 mean zero and variance 1 (the scaling factor is $1/\sqrt{M\ell}$). Let t be the stride length (where $n = (D-\ell)/t + 1$) and build the structured random matrix $\boldsymbol{W}$ as the weight matrix in a single layer CNN for M-channel input dimension D. If*

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

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 have, with high probability, the model-RIP property for $\mathcal{M}_k^2$.*

Other examples of matrices that satisfy model-RIP (both empirically and via a less sophisticated analysis on the dot products between any two columns) include wavelets and localized Fourier bases; both examples that can be easily and efficiently implemented via convolutions in a CNN.

3.2 Reconstruction bounds

To distinguish the true sparse code $\boldsymbol{z}$ and its reconstruction, we use $\hat{\boldsymbol{z}} = \mathbb{M}(\boldsymbol{h}, k) = \mathbb{M}(\boldsymbol{W}\boldsymbol{x}, k)$ for the reconstruction by CNN. Our next result tells us that if we compute the hidden units $\boldsymbol{h}$ from an input signal $\boldsymbol{x}$ using a weight matrix $\boldsymbol{W}$ whose transpose has the model-RIP and using max pooling over each filter ($\hat{\boldsymbol{z}}$), then we can reconstruct (approximately) the input signal $\boldsymbol{x}$ simply by multiplying the hidden units by $\boldsymbol{W}$. This result bounds the relative error between the approximate reconstruction $\hat{\boldsymbol{x}}$ and the input as a function of the distortion for the model-RIP. In our analysis, we assume that the input signal $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$ is a sparse linear combination of hidden activations, captured approximately by the filters in $\boldsymbol{W}$. 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.

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*

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

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.

4 Experimental Evidence and Analysis

In this section, we provide experimental validation of our theoretical model and analysis. We first validate experimentally the relevance of our assumption by examining the effectiveness of random filter CNNs. We then provide an experimental validation of our theoretical analysis on the synthetic 1D case, then we provide experimental results on more realistic scenarios. In particular, we study popular deep neural networks trained for image classification on the ImageNet ILSVRC 2012 dataset (Deng et al., 2009). We calculate empirical model-RIP bounds for $\boldsymbol{W}^T$, showing that they are consistent with theory. Our results are also consistent with a long line of research shows that it is reasonable to model real, natural images as sparse linear combinations over learned dictionaries (e.g., Boureau et al. (2008); Le et al. (2013); Lee et al. (2008); Olshausen et al. (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. (This is the relative ℓ_2 distance between the original image and the reconstruction.) We investigate both randomly sampled filters and empirically learned filters in these experiments. Our implementation is based on the Caffe (Jia et al., 2014) and MatConvNet (Vedaldi and Lenc, 2015) toolboxes.

4.1 Evaluation of 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 dataset. 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. Specifically, we test random CNNs with 1, 2, and 3 convolutional layers, where we use ReLU as the activation. A 2×2 max pooling layer follows each convolutional layer to down-sample the feature map.[8] We experiment with different filter sizes $(3, 5, 7)$ and numbers of channels $(64, 128, 256, 1024, 2048)$ and report the classification accuracy of the best-performing architectures based on 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 Section C.1 of the supplementary materials. We observe the CNNs with Gaussian random filters achieve surprisingly good classification performance (implying that they serve as reasonable representation of input data), although fully learnable CNN counterparts perform better. Our experimental results are also consistent with the observations made by Jarrett et al. (2009) and Saxe et al. (2011). Overall, these results seem to suggest that the 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.

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 Section C.1 in the supplementary materials for the architecture details). The random filters, assumed in our theoretical analysis, perform reasonably well, not far off the learned filters.

4.2 Experimental Validation of the Analysis in 1D Synthetic Data

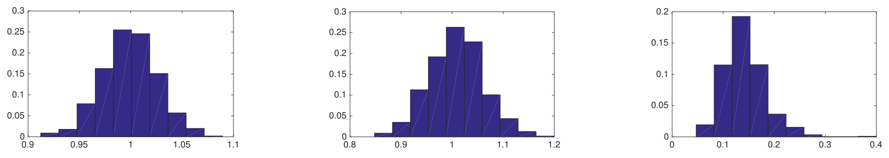
We use 1-D synthetic data to empirically show the basic validity of our theory in terms of the model-RIP condition in Equation (1) and reconstruction bound in Theorem 3.3. We plot the histograms of the empirical model-RIP values of 1D Gaussian random filters $\boldsymbol{W}$ (scaled by $1/\sqrt{lM}$) with size $l \times 1 \times M \times K = 5 \times 1 \times 32 \times 96$ on 1D $\mathcal{M}_k$ 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 2a and 2b are tightly centered around 1, suggesting that $\boldsymbol{W}^T$ satisfies the model-RIP condition in Equation (1) and its corollary from Lemma B.1 in the supplementary materials. We also empirically show the reconstruction bound in Theorem 3.3 on synthetic vectors $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$ (Figure 2c). The reconstruction error is concentrated at around 0.1–0.2 and bound under 0.5. Results in Figure 2 suggests the practical validity of our theory when the model assumptions hold.

4.3 Architectures and Dataset

We conduct the rest of our experimental evaluations on the 16-layer VGGNet (Model D in Simonyan and Zisserman (2015)),[9] 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 ImageNet 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 the supplementary materials.

[8]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. The filter weights of the intermediate layers in the CNNs are not trained after random initialization. On top of the network, we use an optional average pooling layer to reduce the feature map size to 4×4 and a dropout layer for better regularization before feeding the feature to a learnable soft-max classifier for image classification.

[9]VGGNet is practically important as it is popularly used in the community and is one of the best-performing “single-pathway” networks (i.e., no skip connections). We expect that the ResNet (e.g., trained from ImagNet) can also reconstruct images from its activations well in practice. However, the ResNet architectures are too complicated to be in the scope of our theory without further nontrivial customization.



(a) (b) (c)

Figure 2: For 1D scaled Gaussian random filters $\boldsymbol{W}$, we plot the histogram of ratios (a) $\|\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (model-RIP condition in Equation (1); supposed to be concentrated at 1), (b) $\|\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (model-RIP corollary from Lemma B.1 in the supplementary materials; 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 $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\mathbb{M}_{\text{fixed}}(\boldsymbol{W}\boldsymbol{x}, k)$ is the reconstruction of $\boldsymbol{x}$, where we use the naive unsampling to recover the reduced dimension due to pooling (see Section 2.3).

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

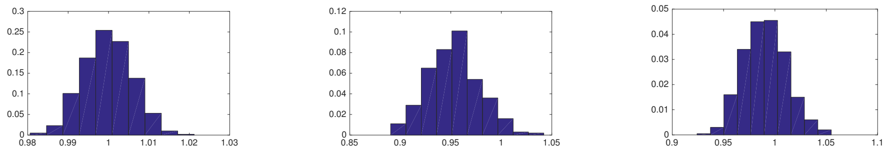
Table 2: Layer-wise sparsity of VGGNet on ILSVRC-2012 validation set. "c" stands for convolutional layers while "p" represents pooling layers. CNN with random filters in Section 4.4 can be simulated with the same sparsity.

VGGNet contains five groups of convolution and pooling layers, each group has 2~3 convolutional layers followed by a pooling layer. We denote the j-th convolutional layer in the i-th group "conv(i, j)," and the pooling layer "pool(i)." When we say the activations/features are from i-th layer, we mean they are the output of pool(i). Our analysis is for single convolutional layers. When evaluating the i-th layer, we take the activations from the $(i-1)$-th layer, and investigate the filters and output of conv$(i, 1)$.

4.4 2D MODEL-RIP

The key to our reconstruction bound is Theorem 3.3 is the model-RIP condition for our particular model of sparsity in Equation (1). We empirically evaluate the model-RIP property, i.e., $\|\boldsymbol{W}^T\boldsymbol{z}\|/\|\boldsymbol{z}\|$, for real CNN filters of the pretrained VGGNet. We use two-dimensional coefficients (or hidden units) $\boldsymbol{z}$ (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 suggest that the sparsity model and the model-RIP property of the filters are consistent with what we conclude from the mathematical analysis on the simpler one-dimensional case.

To check the significance of the model-RIP property (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 (refer to Table 2). Given the 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 property. To be consistent with succeeding experiments, we choose conv$(5, 2)$, while other layers



(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 $\|\boldsymbol{W}^T\boldsymbol{z}\|_2/\|\boldsymbol{z}\|_2$ (the model-RIP value derived from Equation (1); supposed to be concentrated at 1) where $\boldsymbol{z}$ is a $\mathcal{M}_k$ sparse signal. (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. (b) $\boldsymbol{z}$ is recovered by Algorithm 2 from the conv(5,1) activations before applying ReLU. (c) $\boldsymbol{z}$ is recovered by Algorithm 2 from the conv(5,1) activations after applying ReLU. The learned filters admits similar model-RIP value distributions to the random filters except for a bit larger bandwidth, which means the model-RIP condition in Equation (1) can empirically hold even when the filters do not necessarily subject to the i.i.d Gaussian random assumption.

show similar results. Figure 3 (a) summarizes the distribution of empirical model-RIP values, which is clearly centered around 1 and satisfies Equation (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.[10] We set $D = 15$ (the same as the output activations of conv$(5, 2)$) and use 2×2 pooling regions[11] (commonly used in recent deep networks). We generate 1000 $\mathcal{M}_k$ randomly sampled sparse activation ($\boldsymbol{z}$) maps 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.

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: $\boldsymbol{z}^{\text{model}} = \mathbb{M}_{\text{known}}(\boldsymbol{z}^{\text{init}}, k)$
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 $\boldsymbol{z}_i^{\text{model}} = 0$
```

To conduct more realistic experiments, we observe 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 property 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 $\mathbb{M}_{\text{known}}$, defined in (3), 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 property 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 rough less then 0.05, which agrees with the theoretical bound (1) quite well.

To gain more insight, we summarize the learned filter coherence in Table 4 for all the convolutional layers in VGGNet.[12] 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 property turns out to be robust to this mismatch. It also demonstrates the strong invertibility of CNN in practice.

4.5 Reconstruction Bounds

With model-RIP as a sufficient condition, Theorem 3.3 provides theoretical bounds for layer-wise reconstruction via $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\mathbb{M}(\boldsymbol{W}\boldsymbol{x}, k)$. This operator 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 Layer 1~4 activations (i.e., pool(1)~(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 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.

[10]We do not remove any filters including those in approximate positive/negative pairs (refer to 3).

[11]In VGGNet, no pooling layer follows conv$(5, 2)$. Here, we just use it in this way to analyze the convolution-pooling pair targeted by our theory.

[12]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.

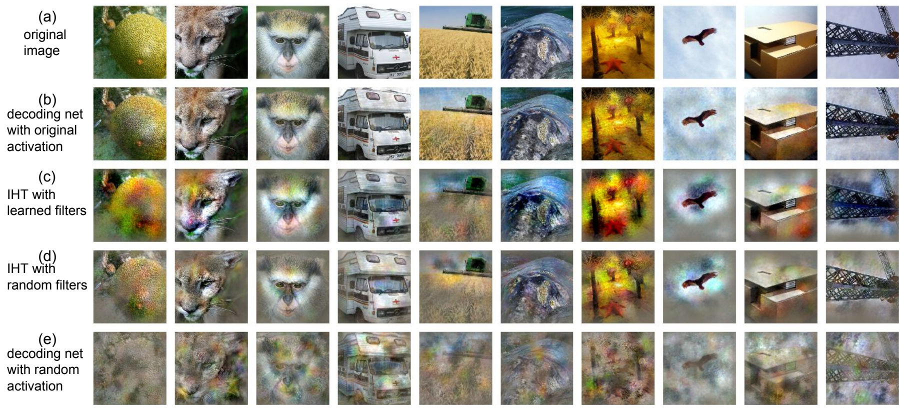

(a) original image
(b) decoding net with original activation
(c) IHT with learned filters
(d) IHT with random filters
(e) decoding net with random activation

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, (e) output of the decoding net with Gaussian random activation.

As an example, Figure 4 illustrates the image reconstruction results for the hidden activations of the 4-th layer, the ground truth of which is obtained by feeding natural images to the CNNs. Interestingly, the decoding network itself is powerful, since it can reconstruct the glimpse of images with Gaussian random input, as shown in Figure 4 (e). Object shapes are recovered 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 4-th layer 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 4). 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 the supplementary materials (Figure 5 in Section C.3).

In Table 3, we summarize reconstruction performance for all 4 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.

Additionally, reconstruction performance of IHT is strongly related to the filter coherence, summarized in Table 4. Lower coherence agrees more closely with the model assumptions and leads to higher reconstruction quality. Higher coherence yields worse recovery of the hidden activation (i.e., large $\|\hat{\boldsymbol{z}} - \boldsymbol{z}\|$, where $\hat{\boldsymbol{z}}$ is the hidden activations recovered by IHT, $\boldsymbol{z}$ is the true activation). Compared to Algorithm 2, (one-step) IHT is not so robust to high coherence.

In summary, when the assumption of i.i.d Gaussian randomness of the CNN filters holds, our theoretical reconstruction bound strictly match with the empirical observations. More importantly, we demonstrate that the bound can still reasonably hold in practice for discriminatively learned CNN layers, which is particularly true for layers with relatively lower coherence.

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

[13]The relative error in activation space of random activations (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 $\|\boldsymbol{f}\| = \|\hat{\boldsymbol{f}}\|$.

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 3: Layer-wise relative reconstruction errors by different methods in activation space and image space between reconstructed and original activations. For 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.[13]

layer	(1,1)	(1,2)	(2,1)	(2,2)	(3,1)	(3,2)	(3,3)
coherence of learned filters	0.9427	0.7340	0.6435	0.7465	0.5838	0.4844	0.5194
coherence of random filters	0.6701	0.1218	0.1546	0.1053	0.1099	0.0895	0.0802

layer	(4,1)	(4,2)	(4,3)	(5,1)	(5,2)	(5,3)
coherence of learned filters	0.4596	0.4574	0.4043	0.4099	0.4099	0.4046
coherence of random filters	0.0920	0.0619	0.0617	0.0696	0.0674	0.0674

Table 4: Comparison of coherence between learned filters in each convolutional layer of VGGNet and Gaussian random filters with corresponding sizes.

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.

References

S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable Bounds for Learning Some Deep Representations. *ICML*, pages 584–592, 2014.

S. Arora, Y. Liang, and T. Ma. Why are deep nets reversible: A simple theory, with implications for training. *arXiv:1511.05653*, 2015.

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

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.

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

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

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

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

S. Boyd. $l1$-norm methods for convex-cardinality problems, ee364b: Convex optimization ii lecture notes, 2014-2015 spring. 2015.

J. Bruna, A. Szlam, and Y. LeCun. Signal recovery from pooling representations. In *ICML*, pages 307–315, 2014.

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

D. Chicco, P. Sadowski, and P. Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference Bioinformatics, Computational Biology, and Health Informatics*, pages 533–540, 2014.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, June 2009.

A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *CVPR*, 2016.

R. Giryes, G. Sapiro, and A. M. Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

K. He, Y. Wang, and J. Hopcroft. A powerful generative model using random weights for the deep image representation. *arXiv preprint arXiv:1606.04801*, 2016.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. 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.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153, 2009.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

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

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.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area v2. In *NIPS*, 2008.

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

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

B. 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.

J. Y. Park, H. L. Yap, C. 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.

A. Paul and S. Venkatasubramanian. Why does Deep Learning work? - A perspective from Group Theory. *arXiv.org*, Dec. 2014.

M. A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *ICML*, pages 1089–1096, 2011.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016.

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

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv.org*, Nov. 2010.

J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, 19(11):2861–2873, 2010.

Y. Zhang, K. Lee, and H. Lee. Augmenting neural networks with reconstructive decoding pathways for large-scale image classification. In *ICML*, 2016.

J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun. Stacked what-where auto-encoders. *arXiv:1506.02351*, 2016.

Supplementary Materials: Towards Understanding the Invertibility of Convolutional Neural Networks

A Mathematical Analysis: Model-RIP and random filters

Theorem 3.1*(Restated) 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 mean zero and variance 1 (the scaling factor is $1/\sqrt{M\ell}$). Let t be the stride length (where $n = (D-\ell)/t + 1$) and build the structured random matrix $\boldsymbol{W}$ as the weight matrix in a single layer CNN for M-channel input dimension D. If*

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

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.

Proof. We note that this result follows the same structure of that for many proofs of the RIP for (structured) random matrices (see Park et al. (2011); Vershynin (2010) for details) although we make minor tweaks to account for the particular structure of $\boldsymbol{W}^T$.

Suppose that $\boldsymbol{z} \in \mathcal{M}_k$ which means that $\boldsymbol{z}$ consists of at most k non-zero entries that each appear in a distinct block of size n (there are a total of K blocks). First, we observe that the norm of $\boldsymbol{W}^T\boldsymbol{z}$ is preserved in expectation.

Lemma A.1.

$$\mathbb{E}(\|\boldsymbol{W}^T\boldsymbol{z}\|_2^2) = \|\boldsymbol{z}\|_2^2$$

Proof. Note that each entry of $\boldsymbol{W}^T$ is either zero or Gaussian random variable $w \sim N(0,1)$ (suitably normalized). Therefore, it is obvious that $\mathbb{E}(\boldsymbol{W}\boldsymbol{W}^T) = \boldsymbol{I}$ since each row of $\boldsymbol{W}$ satisfies $\mathbb{E}\left(\left(\boldsymbol{w}_{i,m_1}^{j_1}\right)^T\left(\boldsymbol{w}_{i,m_2}^{j_2}\right)\right) = 0$ if $j_1 \neq j_2$ or $m_1 \neq m_2$, and we normalized the random variables so that $\mathbb{E}\left(\left\|\left[\left(\boldsymbol{w}_{i,1}^{j}\right)^T, \ldots, \left(\boldsymbol{w}_{i,M}^{j}\right)^T\right]\right\|_2\right) = 1$ for all j. Finally, we have

$$\mathbb{E}\left(\|\boldsymbol{W}^T\boldsymbol{z}\|_2^2\right) = \mathbb{E}\left(\boldsymbol{z}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z}\right) = \boldsymbol{z}^T\mathbb{E}\left(\boldsymbol{W}\boldsymbol{W}^T\right)\boldsymbol{z} = \boldsymbol{z}^T\boldsymbol{z} = \|\boldsymbol{z}\|_2^2.$$

□

Let $\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{z}$. We aim to show that the square norm of the random variable $\|\boldsymbol{y}\|_2^2$ concentrates tightly about its mean; i.e., with exceedingly low probability

$$\Big|\|\boldsymbol{y}\|_2^2 - \|\boldsymbol{z}\|_2^2\Big| > \delta\|\boldsymbol{z}\|_2^2.$$

To do so, we need several properties of sub-Gaussian and sub-exponential random variables. A mean-zero **sub-Gaussian** random variable Z has a moment generating function that satisfies

$$\mathbb{E}(\exp(tZ)) \leq \exp(t^2C^2)$$

for all $t \in \mathbb{R}$ and some constant C. The **sub-Gaussian norm** of Z, denoted $\|Z\|_{\psi_2}$ is

$$\|Z\|_{\psi_2} = \sup_{p\geq 1}\frac{1}{\sqrt{p}}\Big(\mathbb{E}|Z|^p\Big)^{1/p}.$$

If $Z \sim N(0,\sigma^2)$, then $\|Z\|_{\psi_2} \leq c\sigma$.

A **sub-exponential** random variable X satisfies[14]

$$\mathbb{P}\Big(|X| > t\Big) \leq \exp(1 - t/C)$$

[14]There are two other equivalent properties. See Vershynin (2010) for details.

for all $t \geq 0$.

Let $\boldsymbol{y}_i$ denote the ith entry of the vector $\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{z}$. We can write

$$\boldsymbol{y}_i = \sum_{j=1}^{Kn} \boldsymbol{W}_{i,j}^T \boldsymbol{z}_j$$

and observe that $\boldsymbol{y}_i$ is a linear combination of i.i.d. sub-Gaussian random variables (or it is identically equal to 0) and, as such, is itself a sub-Gaussian random variable with mean zero and sub-Gaussian norm $\|\boldsymbol{y}_i\|_{\psi_2} \leq C/\sqrt{M\ell}\|w\|_{\psi_2}\|\boldsymbol{z}\|_2$ (see Vershynin (2010), Lemma 5.9). The structure of the random matrix and how many non-zero entries are in row i of $\boldsymbol{W}$ do enter the more refined bound on the sub-Gaussian norm of $\|\boldsymbol{y}_i\|_{\psi_2}$ (again, see Vershynin (2010), Lemma 5.9 for details) but we ignore such details for this estimate as they are not necessary for the next estimate.

To obtain a concentration bound for $\|\boldsymbol{y}_i\|_2^2$, we recall from Park et al. (2011); Vershynin (2010) that the sum of squares of sub-Gaussian random variables tightly concentrate.

Theorem A.2. *Let $Y_1, \dots, Y_{MD}$ be independent sub-Gaussian random variables with sub-Gaussian norms $\|Y_i\|_{\psi_2}$ for all $i = 1, \dots, MD$. Let $T = \max_i \|Y_i\|_{\psi_2}$. The for every $t \geq 0$ and every $\boldsymbol{a} \in \mathbb{R}^{MD}$,*

$$\mathbb{P}\left(\Big|\sum_{i=1}^{MD} \boldsymbol{a}_i(Y_i - \mathbb{E}Y_i^2)\Big| \geq t\right) \leq 2\exp\left(-C\min\Big(\frac{Ct^2}{T^2\|\boldsymbol{a}\|_2^2}, \frac{Ct}{T\|\boldsymbol{a}\|_\infty}\Big)\right).$$

We note that although some entries $\boldsymbol{y}_i$ may be identically zero, depending on the sparsity pattern of $\boldsymbol{z}$, not all entries are. Let us define $\tilde{\boldsymbol{y}}_i = \frac{\boldsymbol{y}_i}{\|\boldsymbol{y}_i\|_{\psi_2}}$ so that $\|\tilde{\boldsymbol{y}}_i\|_{\psi_2} = 1$ and observe that

$$\mathbb{P}\Big(\Big|\|\boldsymbol{y}\|_2^2 - \|\boldsymbol{z}\|_2^2\Big| > \delta\|\boldsymbol{z}\|_2^2\Big) = \mathbb{P}\left(\Big|\sum_{i=1}^{MD} \|\boldsymbol{y}_i\|_{\psi_2}^2(\tilde{\boldsymbol{y}}_i^2 - \mathbb{E}\tilde{\boldsymbol{y}}_i^2)\Big| > \delta\|\boldsymbol{z}\|_2^2\right).$$

We apply Theorem A.2 to the sub-Gaussian random variables $\tilde{\boldsymbol{y}}_i$ with the weights $\|\boldsymbol{y}_i\|_{\psi_2}^2$. We have

$$\|\boldsymbol{a}\|_2^2 = \sum_{i=1}^{MD} \|\boldsymbol{y}_i\|_{\psi_2}^4 \leq \frac{CD\|w\|_{\psi_2}^4\|\boldsymbol{z}\|_2^4}{M\ell^2} \quad \text{and} \quad \|\boldsymbol{a}\|_\infty \leq \frac{C\|w\|_{\psi_2}^2\|\boldsymbol{z}\|_2^2}{M\ell}.$$

If we set $T = 1$, $t = \delta\|\boldsymbol{z}\|_2^2$, and use the above estimates for the norms of $\boldsymbol{a}$, we have

$$\mathbb{P}\Big(\Big|\|\boldsymbol{y}\|_2^2 - \|\boldsymbol{z}\|_2^2\Big| > \delta\|\boldsymbol{z}\|_2^2\Big) \leq 2\exp\left(-C\min\Big(\frac{C\delta^2 M\ell^2}{D\|w\|_{\psi_2}^4}, \frac{C\delta M\ell}{\|w\|_{\psi_2}^2}\Big)\right). \tag{4}$$

Finally, we use the concentration of measure result in a crude union bound to bound the failure probability over all vectors $\boldsymbol{z} \in \mathcal{M}_k$. We take $n^k\binom{K}{k} \approx (nK)^k$ and ϵ for a desired constant failure probability. Using the smaller term in Equation (4), (note that $\delta < 1$, $\ell/D < 1$, and $\|w\|_{\psi_2} \geq 1$) we have

$$\exp\Big(-C\frac{M\ell^2\delta^2}{D\|w\|_{\psi_2}^4}\Big)\exp\Big(k(\log(K) + \log(n))\Big) \leq \exp(\log(\epsilon))$$

which implies

$$\frac{M\ell^2}{D} \geq \frac{k}{\delta^2}\|w\|_{\psi_2}^4\left(\log(K) + \log(n) - \log(\epsilon)\right) = C\frac{k}{\delta^2}\Big(\log(K) + \log(n) - \log(\epsilon)\Big).$$

Therefore, if design our matrix $\boldsymbol{W}$ as described and with the parameter relationship as above, the matrix $\boldsymbol{W}^T$ with satisfy the model-RIP for $\mathcal{M}_k$ and parameter δ with probability $1 - \epsilon$. □

Let us discuss the relationship amongst the parameters in our result. First, if we have only one channel $M = 1$ and the filter length $\ell = D$, then our bound on the number of measurements D matches those of traditional (model-based) compressive sensing; namely,

$$D \geq C\frac{k}{\delta^2}\left(\log(K) + \log(n) - \log(\epsilon)\right).$$

If $\ell < D$ (i.e., the filters are much shorter than the length of the input signal as in a CNN), then we can compensate by adding more channels; i.e., the filter length ℓ needs to be larger than $\sqrt{D}$, or, if add more channels, $\sqrt{D/M}$.

B Mathematical Analysis: Reconstruction Bounds

The consequences of having model-RIP are two-fold. The first is that if we assume that an input image is the structured sparse linear combination of filters, $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}$ (where $\boldsymbol{z} \in \mathcal{M}_k$ and $\boldsymbol{W}^T$ satisfies the model-RIP property), then we know an upper and lower bound on the norm of $\boldsymbol{x}$ in terms of the norm of its sparse coefficients, $\|\boldsymbol{x}\|_2 \leq (1 \pm \delta)\|\boldsymbol{z}\|_2$. Additionally,

$$\|\boldsymbol{z}\|_2 \leq \frac{1}{\sqrt{1-\delta}}\|\boldsymbol{x}\|_2.$$

More importantly, when we calculate the hidden units of $\boldsymbol{x}$,

$$\boldsymbol{h} = \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x}) = \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{z})$$

we can see that the computation of $\boldsymbol{h}$ is nothing other than the first step of a reconstruction algorithm analogous to that of model-based compressed sensing. As a result, we have a bound on the error between $\boldsymbol{h}$ and $\boldsymbol{z}$ and we see that we can analyze the approximation properties of a feedfoward CNN and its linear reconstruction algorithm. In particular, we can conclude that a feedforward CNN and a linear reconstruction algorithm provide a good approximation to the original input image.

Theorem 3.3*(Restated) 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*

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

Proof. To show this result, we recall the two following lemmas from Baraniuk et al. (2010) and rephrase them in the setting of a feedforward CNN.

Lemma B.1. *Suppose $\boldsymbol{W}^T$ has $\mathcal{M}_k$-RIP with constant δ_k. Let Ω be a support corresponding to a subspace in $\mathcal{M}_k$. Then we have the following bounds:*

$$\|\boldsymbol{W}_\Omega\boldsymbol{x}\|_2 \leq \sqrt{1+\delta_k}\|\boldsymbol{x}\|_2 \tag{5}$$

$$\|\boldsymbol{W}_\Omega\boldsymbol{W}_\Omega^T\boldsymbol{z}\|_2 \leq (1+\delta_k)\|\boldsymbol{z}\|_2 \tag{6}$$

$$\|\boldsymbol{W}_\Omega\boldsymbol{W}_\Omega^T\boldsymbol{z}\|_2 \geq (1-\delta_k)\|\boldsymbol{z}\|_2 \tag{7}$$

Lemma B.2. *Suppose that $\boldsymbol{W}^T$ has $\mathcal{M}_k^2$-RIP with constant δ_{2k}. Let Ω be a support corresponding to a subspace of $\mathcal{M}_k$ and suppose that $\boldsymbol{z} \in \mathcal{M}_k$ (not necessarily supported on Ω). Then*

$$\|\boldsymbol{W}_\Omega\boldsymbol{W}^T\boldsymbol{z}|_{\Omega^c}\|_2 \leq \delta_{2k}\|\boldsymbol{z}|_{\Omega^c}\|_2.$$

Let Π denote the support of the $\mathcal{M}_k$ sparse vector $\boldsymbol{z}$. Set $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x}$ and set $\hat{\boldsymbol{z}}$ to be the result of max pooling applied to the vector $\boldsymbol{h}$, or the best fit (with respect to the ℓ_2 norm) to $\boldsymbol{h}$ in the model $\mathcal{M}_k$. Let Ω denote the support set of $\hat{\boldsymbol{z}} \in \mathcal{M}_k$. For simplicity, we assume $|\Pi| = k = |\Omega|$.

Lemma B.3 (Identification)**.** *The support set, Ω, of the switch units captures a significant fraction of the total energy in the coefficient vector $\boldsymbol{z}$*

$$\|\boldsymbol{z}|_{\Omega^c}\|_2 \leq \frac{2\delta_{2k}}{1-\delta_k}\|\boldsymbol{z}\|_2.$$

Proof. Let $\boldsymbol{h}_\Omega$ and $\boldsymbol{h}_\Pi$ be the vector $\boldsymbol{h}$ restricted to the support sets Ω and Π, respectively. Since both are support sets for $\mathcal{M}_k$ and since Ω is the best support set for $\boldsymbol{h}$,

$$\|\boldsymbol{h} - \boldsymbol{h}_\Omega\|_2 \leq \|\boldsymbol{h} - \boldsymbol{h}_\Pi\|_2,$$

and, after several calculations, we have

$$\|\boldsymbol{h}|_{\Omega\setminus\Pi}\|_2^2 \geq \|\boldsymbol{h}|_{\Pi\setminus\Omega}\|_2^2.$$

Using Lemma B.2 and the size $|(\Omega \setminus \Pi) \bigcup \Pi| \leq 2k$, we have

$$\|\boldsymbol{h}_{\Omega\setminus\Pi}\|_2 = \|\boldsymbol{W}_{\Omega\setminus\Pi}\boldsymbol{W}^T\boldsymbol{z}\|_2 \leq \delta_{2k}\|\boldsymbol{z}\|_2.$$

We can bound the other side of the inequality as

$$\begin{aligned}\|\boldsymbol{h}_{\Pi\setminus\Omega}\|_2 &\geq \|\boldsymbol{W}_{\Pi\setminus\Omega}(\boldsymbol{W}^T\boldsymbol{z}|_{\Pi\setminus\Omega})\|_2 - \|\boldsymbol{W}_{\Pi\setminus\Omega}(\boldsymbol{W}^T\boldsymbol{z}|_{\Omega})\|_2 \\ &\geq (1-\delta_k)\|\boldsymbol{z}|_{\Pi\setminus\Omega}\|_2 - \delta_{2k}\|\boldsymbol{z}|_{\Omega}\|_2.\end{aligned}$$

Since the support of $\boldsymbol{z}$ is the set Π, $\Pi \setminus \Omega = \Omega^c$ and we can conclude that

$$\delta_{2k}\|\boldsymbol{z}\|_2 \geq (1-\delta_k)\|\boldsymbol{z}|_{\Omega^c}\|_2 - \delta_{2k}\|\boldsymbol{z}|_{\Omega}\|_2,$$

and with some rearrangement, we have

$$\|\boldsymbol{z}|_{\Omega^c}\|_2 \leq \frac{2\delta_{2k}}{1-\delta_k}\|\boldsymbol{z}\|_2.$$

□

To set the value of $\hat{\boldsymbol{z}}$ on its support set Ω, we simply set $\hat{\boldsymbol{z}} = \boldsymbol{h}|_\Omega$ and $\hat{\boldsymbol{z}}|_{\Omega^c} = 0$. Then

Lemma B.4 (Estimation)**.**

$$\|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_2 \leq \frac{5\delta_{2k}}{1-\delta_k}\|\boldsymbol{z}\|_2$$

Proof. First, note that $\|\boldsymbol{I} - \boldsymbol{W}_\Omega\boldsymbol{W}_\Omega^T\|_2 \leq \delta_k$ since

$$(1-\delta_k) \leq \sup_{\|\boldsymbol{z}\|\neq 0} \frac{\|\boldsymbol{W}_\Omega^T\boldsymbol{z}\|_2^2}{\|\boldsymbol{z}\|_2^2} \left(= \sigma_{\max}^2(\boldsymbol{W}_\Omega^T) = \sigma_{\max}(\boldsymbol{W}_\Omega\boldsymbol{W}_\Omega^T)\right) \leq (1+\delta_k),$$

where $\sigma_{\max}$ is the maximum singular value. Therefore,

$$\begin{aligned}\|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_2 &\leq \|\boldsymbol{z}|_{\Omega^c}\|_2 + \|\boldsymbol{z}|_\Omega - \hat{\boldsymbol{z}}|_\Omega\|_2 \\ &= \|\boldsymbol{z}|_{\Omega^c}\|_2 + \|\boldsymbol{z}|_\Omega - \boldsymbol{W}_\Omega(\boldsymbol{W}^T\boldsymbol{z}|_\Omega + \boldsymbol{W}^T\boldsymbol{z}|_{\Omega^c})\|_2 \\ &\leq \|\boldsymbol{z}|_{\Omega^c}\|_2 + \|(\boldsymbol{I} - \boldsymbol{W}_\Omega\boldsymbol{W}_\Omega^T)\boldsymbol{z}|_\Omega\|_2 + \|\boldsymbol{W}_\Omega\boldsymbol{W}^T\boldsymbol{z}|_{\Omega^c}\|_2 \\ &\leq \|\boldsymbol{z}|_{\Omega^c}\|_2 + \|\boldsymbol{I} - \boldsymbol{W}_\Omega\boldsymbol{W}_\Omega^T\|_2\|\boldsymbol{z}|_\Omega\|_2 + \delta_{2k}\|\boldsymbol{z}|_{\Omega^c}\|_2 \\ &\leq \|\boldsymbol{z}|_{\Omega^c}\|_2 + \delta_k\|\boldsymbol{z}|_\Omega\|_2 + \delta_{2k}\|\boldsymbol{z}|_{\Omega^c}\|_2 \\ &\leq \left((1+\delta_{2k})\frac{2\delta_{2k}}{1-\delta_k} + \delta_k\right)\|\boldsymbol{z}\|_2 \\ &\leq \frac{5\delta_{2k}}{1-\delta_k}\|\boldsymbol{z}\|_2.\end{aligned}$$

□

Finally, if we use the autoencoder formulation to reconstruct the original image $\boldsymbol{x}$ by setting $\hat{\boldsymbol{x}} = \boldsymbol{W}^T\hat{\boldsymbol{z}}$, we can estimate the reconstruction error. We note that $\hat{\boldsymbol{z}}$ is $\mathcal{M}_k$-sparse by construction and remind the reader that $\boldsymbol{W}^T$ satisfies $\mathcal{M}_k^2$-model-RIP with constants $\delta_k \leq \delta_{2k} \ll 1$. Then, using Lemma B.4 as well as the $\mathcal{M}_k^2$-sparse properties of $\boldsymbol{W}^T$,

$$\begin{aligned}\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 &= \|\boldsymbol{W}^T(\boldsymbol{z} - \hat{\boldsymbol{z}})\|_2 \leq \sqrt{1+\delta_{2k}}\|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_2 \\ &\leq \frac{5\delta_{2k}}{1-\delta_k}\sqrt{1+\delta_{2k}}\|\boldsymbol{z}\|_2 \\ &\leq \frac{5\delta_{2k}}{1-\delta_k}\frac{\sqrt{1+\delta_{2k}}}{\sqrt{1-\delta_{2k}}}\|\boldsymbol{x}\|_2.\end{aligned}$$

This proves that a feedforward CNN with a linear reconstruction algorithm is an approximate autoencoder and bounds the reconstruction error of the input image in terms of the geometric properties of the filters. □

C More experimental results

C.1 More details on evaluation of CNNs with Gaussian random filters

In this section, we provide more details on the network architectures that we used in Table 1. In particular, we describe the best performing architectures for all cases in Table 5.

Method	#Layers	1 layers	2 layers	3 layers
Random filters	Best param.	(2048)5c-2p_{max}-4p_{ave}	(2048)3c-2p_{max}-(2048)3c-2p_{max}-2p_{ave}	(2048)3c-2p_{max}-(2048)3c-2p_{max}-(1024)3c-2p_{max}
	Accuracy	66.5%	74.6%	74.8%
Learned filters	Best param.	(1024)5c-2p_{max}-4p_{ave}	(1024)3c-2p_{max}-(1024)3c-2p_{max}-2p_{ave}	(1024)3c-2p_{max}-(1024)3c-2p_{max}-(1024)3c-2p_{max}
	Accuracy	68.1%	83.3%	89.3%

Table 5: Best-performing architecture and classification accuracy of random CNNs on CIFAR-10. "([n])[k]c" denotes a convolution layer with a stride 1, a kernel size [k] and [n] output channels, "[k]p_{max}" denotes a max pooling layer with a kernel size [k] and a stride [k], and "[k]p_{ave}" denotes a average pooling layer. A typical layer consists of four operations, namely convolution, ReLU, batch normalization, and max pooling.

C.2 Layer-wise coherence and sparsity for AlexNet

We present coherence (see Table 6) and sparsity level (see Table 7) for each layer in AlexNet.

layer	1	2	3	4	5
coherence of learned filters	0.9172	0.6643	0.6200	0.6382	0.3390
coherence of random filters	0.1996	0.1263	0.0929	0.1073	0.1026

Table 6: Comparison of coherence between learned filters in each layer of AlexNet and Gaussian random filters with corresponding sizes.

layer	conv1	pool1	conv2	pool2	conv3	conv4	conv5	pool5
% of non-zeros	49.41	87.79	18.97	44.13	31.08	30.95	9.78	28.15

Table 7: Layer-wise sparsity of AlexNet on ILSVRC-2012 validation set.

C.3 Visualization of image reconstruction for VGGNet

In Figure 5, we show reconstructed images from each layer using different reconstruction methods via a pretrained decoding network.

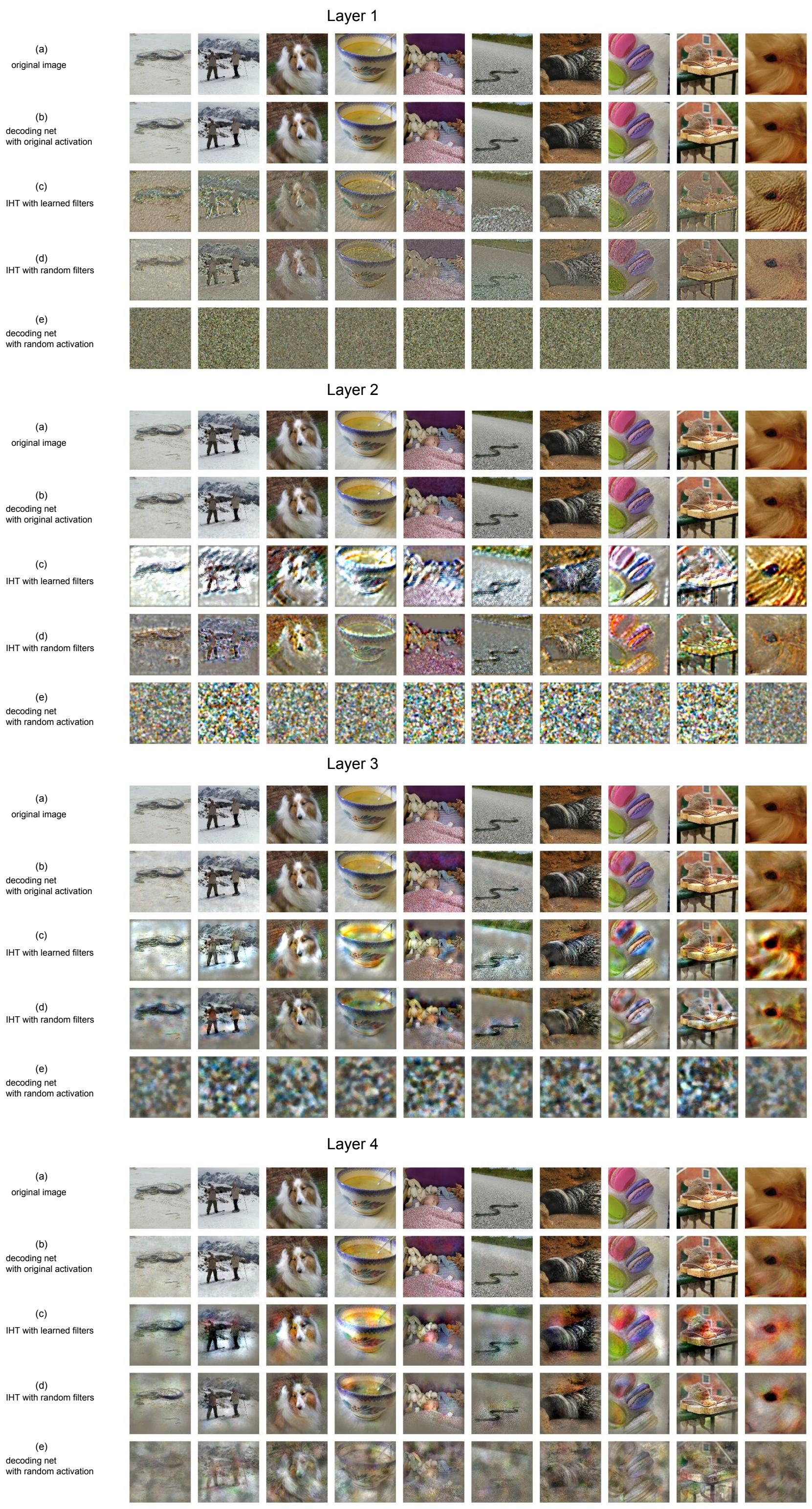

Layer 1
(a) original image
(b) decoding net with original activation
(c) IHT with learned filters
(d) IHT with random filters
(e) decoding net with random activation
Layer 2
(a) original image
(b) decoding net with original activation
(c) IHT with learned filters
(d) IHT with random filters
(e) decoding net with random activation
Layer 3
(a) original image
(b) decoding net with original activation
(c) IHT with learned filters
(d) IHT with random filters
(e) decoding net with random activation
Layer 4
(a) original image
(b) decoding net with original activation
(c) IHT with learned filters
(d) IHT with random filters
(e) decoding net with random activation

Figure 5: 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, (e) output of the decoding net with Gaussian random activation.