Deconvolutional Networks

Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor and Rob Fergus
Dept. of Computer Science, Courant Institute, New York University
{zeiler,dilip,gwtaylor,fergus}@cs.nyu.edu

Abstract

Building robust low and mid-level image representations, beyond edge primitives, is a long-standing goal in vision. Many existing feature detectors spatially pool edge information which destroys cues such as edge intersections, parallelism and symmetry. We present a learning framework where features that capture these mid-level cues spontaneously emerge from image data. Our approach is based on the convolutional decomposition of images under a sparsity constraint and is totally unsupervised. By building a hierarchy of such decompositions we can learn rich feature sets that are a robust image representation for both the analysis and synthesis of images.

1. Introduction

In this paper we propose *Deconvolutional Networks*, a framework that permits the unsupervised construction of hierarchical image representations. These representations can be used for both low-level tasks such as denoising, as well as providing features for object recognition. Each level of the hierarchy groups information from the level beneath to form more complex features that exist over a larger scale in the image. Our grouping mechanism is sparsity: by encouraging parsimonious representations at each level of the hierarchy, features naturally assemble into more complex structures. However, as we demonstrate, sparsity itself is not enough – it must be deployed within the correct architecture to have the desired effect. We adopt a convolutional approach since it provides stable latent representations at each level which preserve locality and thus facilitate the grouping behavior. Using the same parameters for learning each layer, our Deconvolutional Network (DN) can automatically extract rich features that correspond to mid-level concepts such as edge junctions, parallel lines, curves and basic geometric elements, such as rectangles. Remarkably, some of them look very similar to the mid-level tokens posited by Marr in his primal sketch theory [18] (see Fig. 1).

Our proposed model is similar in spirit to the Convolutional Networks of LeCun *et al.* [13], but quite different in operation. Convolutional networks are a bottom-up approach where the input signal is subjected to multiple layers of convolutions, non-linearities and sub-sampling. By contrast, each layer in our Deconvolutional Network is top-down; it seeks to generate the input signal by a sum over convolutions of the feature maps (as opposed to the input) with learned filters. Given an input and a set of filters, inferring the feature map activations requires solving a multi-component deconvolution problem that is computationally challenging. In response, we use a range of tools from low-level vision, such as sparse image priors and efficient algorithms for image deblurring. Correspondingly, our paper is an attempt to link high-level object recognition with low-level tasks like image deblurring through a unified architecture.

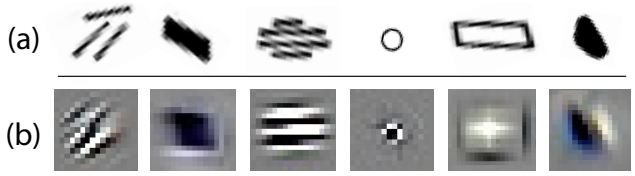

Figure 1. **(a)**: "Tokens" from Fig. 2-4 of *Vision* by D. Marr [18]. These idealized local groupings are proposed as an intermediate level of representation in Marr's primal sketch theory. **(b)**: Selected filters from the 3rd layer of our Deconvolutional Network, trained in an unsupervised fashion on real-world images.

2. Related Work

Deconvolutional Networks are closely related to a number of "deep learning" methods [2, 8] from the machine learning community that attempt to extract feature hierarchies from data. Deep Belief Networks (DBNs) [8] and hierarchies of sparse auto-encoders [22, 9, 26], like our approach, greedily construct layers from the image upwards in an unsupervised fashion. In these approaches, each layer consists of an encoder and decoder[1]. The encoder provides a bottom-up mapping from the input to latent feature space while the decoder maps the latent features back to the input

[1]Convolutional networks can be regarded as a hierarchy of encoder-only layers [13].

space, hopefully giving a reconstruction close to the original input. Going from the input directly to the latent representation without using the encoder is difficult because it requires solving an inference problem (multiple elements in the latent features are competing to explain each part of the input). As these models have been motivated to improve high-level tasks like recognition, an encoder is needed to perform fast, but highly approximate, inference to compute the latent representation at test time. However, during training the latent representation produced by performing top-down inference with the decoder is constrained to be close to the output of the encoder. Since the encoders are typically simple non-linear functions, they have the potential to significantly restrict the latent representation obtainable, producing sub-optimal features. Restricted Boltzmann Machines (RBM), the basic module of DBNs, have the additional constraint that the encoder and decoder must share weights. In Deconvolutional Networks, there is no encoder: we directly solve the inference problem by means of efficient optimization techniques. The hope is that by computing the features exactly (instead of approximately with an encoder) we can learn superior features.

Most deep learning architectures are not convolutional, but recent work by Lee *et al.* [15] demonstrated a convolutional RBM architecture that learns high-level image features for recognition. This is the most similar approach to our Deconvolutional Network, with the main difference being that we use a decoder-only model as opposed to the symmetric encoder-decoder of the RBM.

Our work also has links to recent work in sparse image decompositions, as well as hierarchical representations. Lee *et al.* [14] and Mairal *et al.* [16, 17] have proposed efficient schemes for learning sparse over-complete decompositions of image patches [19], using a convex ℓ_1 sparsity term. Our approach differs in that we perform sparse decomposition over the whole image at once, not just for small image patches. As demonstrated by our experiments, this is vital if rich features are to be learned. The key to making this work efficiently is to use a convolutional approach.

A range of hierarchical image models have been proposed. Particularly relevant is the work of Zhu and colleagues [31, 25], in particular Guo *et al.* [7]. Here, edges are composed using a hand-crafted set of image tokens into large-scale image structures. Grouping is performed via basis pursuit with intricate splitting and merging operations on image edges. The stochastic image grammars of Zhu and Mumford [31] also use fixed image primitives, as well as a complex Markov Chain Monte-Carlo (MCMC)-based scheme to parse scenes. Our work differs in two important ways: first, we learn our image tokens completely automatically. Second, our inference scheme is far simpler than either of the above frameworks.

Zhu *et al.* [30] propose a top-down parts-and-structure model but it only reasons about image edges, as provided by a standard edge detector, unlike ours which directly operates on pixels. The biologically inspired HMax model of Serre *et al.* [24, 23] use exemplar templates in their intermediate representations, rather than learning conjunctions of edges as we do. Fidler and Leonardis [5, 4] propose a top-down model for object recognition which has an explicit notion of parts whose correspondence is explicitly reasoned about at each level. In contrast, our approach simply performs a low-level deconvolution operation at each level, rather than attempting to solve a correspondence problem. Amit and Geman [1] and Jin and Geman [10] apply hierarchical models to deformed Latex digits and car license plate recognition.

3. Model

We first consider a single Deconvolutional Network layer applied to an image. This layer takes as input an image y^i, composed of K_0 color channels $y^i_1, \ldots, y^i_{K_0}$. We represent each channel c of this image as a linear sum of K_1 latent feature maps z^i_k convolved with filters $f_{k,c}$:

$$\sum_{k=1}^{K_1} z^i_k \oplus f_{k,c} = y^i_c \tag{1}$$

Henceforth, unless otherwise stated, symbols correspond to matrices. If y^i_c is an $N_r \times N_c$ image and the filters are $H \times H$, then the latent feature maps are $(N_r + H - 1) \times (N_c + H - 1)$ in size. But Eqn. 1 is an under-determined system, so to yield a unique solution we introduce a regularization term on z^i_k that encourages sparsity in the latent feature maps. This gives us an overall cost function of the form:

$$C_1(y^i) = \frac{\lambda}{2} \sum_{c=1}^{K_0} \| \sum_{k=1}^{K_1} z^i_k \oplus f_{k,c} - y^i_c \|_2^2 + \sum_{k=1}^{K_1} |z^i_k|^p \tag{2}$$

where we assume Gaussian noise on the reconstruction term and some sparse norm p for the regularization. Note that the sparse norm $|w|^p$ is actually the p-norm on the vectorized version of matrix w, i.e. $|w|^p = \sum_{i,j} |w(i,j)|^p$. Typically, $p = 1$, although other values are possible, as described in Section 3.2. λ is a constant that balances the relative contributions of the reconstruction of y^i and the sparsity of the feature maps z^i_k.

Note that our model is top-down in nature: given the latent feature maps, we can synthesize an image. But unlike the sparse auto-encoder approach of Ranzato *et al.* [21], or DBNs [8], there is no mechanism for generating the feature maps from the input, apart from minimizing the cost function C_1 in Eqn. 2. Many approaches focus on bottom-up inference, but we concentrate on obtaining high quality latent representations.

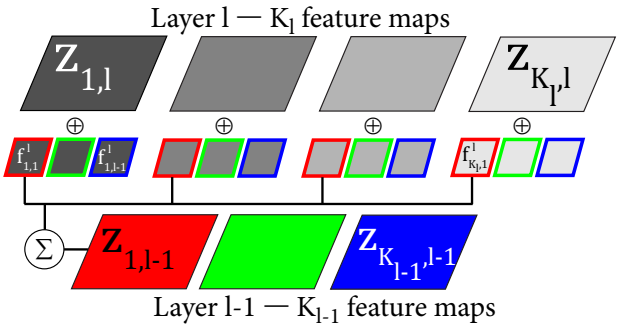

Figure 2. A single Deconvolutional Network layer (best viewed in color). For clarity, only the connectivity for a single input map is shown. In practice the first layer is fully connected, while the connectivity of the higher layers is specified by the map g^l, which is sparse.

In learning, described in Section 3.2, we use a set of images $y = \{y^1, \ldots, y^I\}$ for which we seek $\text{argmin}_{f,z} C_1(y)$[2], the latent feature maps for each image and the filters. Note that each image has its own set of feature maps while the filters are common to all images.

3.1. Forming a hierarchy

The architecture described above produces sparse feature maps from a multi-channel input image. It can easily be stacked to form a hierarchy by treating the feature maps $z^i_{k,l}$ of layer l as input for layer $l+1$. In other words, layer l has as its input an image with K_{l-1} channels being the number of feature maps at layer $l-1$. The cost function C_l for layer l is a generalization of Eqn. 2, being:

$$C_l(y) = \frac{\lambda}{2} \sum_{i=1}^{I} \sum_{c=1}^{K_{l-1}} \| \sum_{k=1}^{K_l} g^l_{k,c} (z^i_{k,l} \oplus f^l_{k,c}) - z^i_{c,l-1} \|_2^2 + \sum_{i=1}^{I} \sum_{k=1}^{K_l} |z^i_{k,l}|^p \quad (3)$$

where $z^i_{c,l-1}$ are the feature maps from the previous layer, and $g^l_{k,c}$ are elements of a fixed binary matrix that determines the connectivity between the feature maps at successive layers, i.e. whether $z^i_{k,l}$ is connected to $z^i_{c,l-1}$ or not [13]. In layer-1 we assume that $g^1_{k,c}$ is always 1, but in higher layers it will be sparse. We train the hierarchy from the bottom upwards, thus $z^i_{c,l-1}$ is given from the results of learning on $C_{l-1}(y)$. This structure is illustrated in Fig. 2.

Unlike several other hierarchical models [15, 21, 9] we do not perform any pooling, sub-sampling or divisive normalization operations between layers, although they could easily be incorporated.

[2]We define $C_1(y) = \sum_i C_1(y^i)$

3.2. Learning filters

To learn the filters, we alternately minimize $C_l(y)$ over the feature maps while keeping the filters fixed (i.e. perform inference) and then minimize $C_l(y)$ over the filters while keeping the feature maps fixed. This minimization is done in a layer-wise manner starting with the first layer where the inputs are the training images y. Details are given in Algorithm 1. We now describe how we learn the feature maps and filters by introducing a framework suited for large scale problems.

Inferring feature maps: Inferring the optimal feature maps $z^i_{k,l}$, given the filters and inputs is the crux of our approach. The sparsity constraint on $z^i_{k,l}$ which prevents the model from learning trivial solutions such as the identity function. When $p = 1$ the minimization problem for the feature maps is convex and a wide range of techniques have been proposed [3, 14]. Although in theory the global minimum can always be found, in practice this is difficult as the problem is very poorly conditioned. This is due to the fact that elements in the feature maps are coupled to one another through the filters. One element in the map can be affected by another distant element, meaning that the minimization can take a very long time to converge to a good solution.

We tried a range of different minimization approaches to solve Eqn. 3, including direct gradient descent, Iterative Reweighted Least Squares (IRLS) and stochastic gradient descent. We found that direct gradient descent suffers from the usual problem of flat-lining and thereby gives a poor solution. IRLS is too slow for large-scale problems with many input images. Stochastic gradient descent was found to require many thousands of iterations for convergence.

Instead, we introduce a more general framework that is suitable for any value of $p > 0$, including pseudo-norms where $p < 1$. The approach is a type of continuation method, as used by Geman [6] and Wang *et al.* [27]. Instead of optimizing Eqn. 3 directly, we minimize an auxiliary cost function $\hat{C}_l(y)$ which incorporates auxiliary variables $x^i_{k,l}$ for each element in the feature maps $z^i_{k,l}$:

$$\hat{C}_l(y) = \frac{\lambda}{2} \sum_{i=1}^{I} \sum_{c=1}^{K_{l-1}} \| \sum_{k=1}^{K_l} g^l_{k,c} (z^i_{k,l} \oplus f^l_{k,c}) - z^i_{c,l-1} \|_2^2 + \frac{\beta}{2} \sum_{i=1}^{I} \sum_{k=1}^{K_l} \| z^i_{k,l} - x^i_{k,l} \|_2^2 + \sum_{i=1}^{I} \sum_{k=1}^{K_l} |x^i_{k,l}|^p \quad (4)$$

where β is a continuation parameter. Introducing the auxiliary variables separates the convolution part of the cost function from the $|\cdot|^p$ term. By doing so, an alternating form of minimization for $z^i_{k,l}$ can be used. We first fix $x^i_{k,l}$ yielding a quadratic problem in $z^i_{k,l}$. Then, we fix $z^i_{k,l}$ and solve a separable 1D problem for each element in $x^i_{k,l}$. We call these two stages the z and x sub-problems respectively.

As we alternate between these two steps, we slowly increase β from a small initial value until it strongly clamps $z^i_{k,l}$ to $x^i_{k,l}$. This has the effect of gradually introducing the sparsity constraint and gives good numerical stability in practice [11, 27]. We now consider each sub-problem.

z **sub-problem**: From Eqn. 4, we see that we can solve for each $z^i_{k,l}$ independently of the others. Here we take derivatives of $\hat{C}_l(y)$ w.r.t. $z^i_{k,l}$, assuming a fixed $x^i_{k,l}$:

$$\frac{\partial \hat{C}_l(y)}{\partial z^i_{k,l}} = \lambda \sum_{c=1}^{K_{l-1}} F^{l^T}_{k,c}(\sum_{\tilde{k}=1}^{K} F^l_{\tilde{k},c} z^i_{k,l} - z^i_{c,l-1}) + \beta(z^i_{k,l} - x^i_{k,l}) \tag{5}$$

where if $g^l_{k,c} = 1$, $F^l_{k,c}$ is a sparse convolution matrix[3] equivalent to convolving with $f^l_{k,c}$, and is zero if $g^l_{k,c} = 0$. Although a variety of other sparse decomposition techniques [16, 21] use stochastic gradient descent methods to update $z^i_{k,l}$ for each i, k separately, this is not viable in a convolutional setting. Here, the various feature maps compete with each other to explain local structure in the most compact way. This requires us to simultaneously optimize over all $z^i_{k,l}$'s for a fixed i and varying k. For a fixed i, setting $\frac{\partial \hat{C}(y)}{\partial z^i_{k,l}} = 0 \;\; \forall\, k$, the optimal $z^i_{k,l}$ are the solution to the following $K_l(N_r + H - 1)(N_c + H - 1)$linear system:

$$A \begin{pmatrix} z^i_{1,l} \\ \cdot \\ z^i_{K_l,l} \end{pmatrix} = \begin{pmatrix} \sum_{c=1}^{K_{l-1}} F^{l^T}_{1,c} z^i_{c,l-1} + \frac{\beta}{\lambda} x^i_{1,l} \\ \cdot \\ \sum_{c=1}^{K_{l-1}} F^{l^T}_{K,c} z^i_{c,l-1} + \frac{\beta}{\lambda} x^i_{K_l,l} \end{pmatrix} \tag{6}$$

where

$$A = \begin{pmatrix} \sum_{c=1}^{K_{l-1}} F^{l^T}_{1,c} F^l_{1,c} + \frac{\beta}{\lambda} I & \cdot & \sum_{c=1}^{K_{l-1}} F^l_{1,c} F^{l^T}_{K_l,c} \\ \cdot & \cdot & \cdot \\ \sum_{c=1}^{K_{l-1}} F^{l^T}_{K_l,c} F^l_{1,c} & \cdot & \sum_{c=1}^{K_{l-1}} F^{l^T}_{K_l,c} F^l_{K_l,c} + \frac{\beta}{\lambda} I \end{pmatrix} \tag{7}$$

In the above equation, $x^i_{1,l}, \ldots, x^i_{K_l,l}$, $z^i_{c,l-1}$ and $z^i_{1,l}, \ldots, z^i_{K_l,l}$ are in vectorized form. Eqn. 6 can be effectively minimized by conjugate gradient (CG) descent. Note that A never needs to be formed since the Az product can be directly computed using convolution operations inside the CG iteration. Each Az product requires $2cK_l$ convolutions of filters with the $(N_r + H - 1)(N_c + H - 1)$ filter maps and can easily be parallelized.

Although some speed-up might be gained by using FFTs in place of spatial convolutions, particularly if the filter size H is large, this can introduce boundary effects in the feature maps – therefore solving in the spatial domain is preferred.

x **sub-problem**: Given fixed $z^i_{k,l}$, finding the optimal $x^i_{k,l}$ requires solving a 1D optimization problem for each element in the feature map. If $p = 1$ then, following Wang *et al.* [27], $x^i_{k,l}$ has a closed-form solution given by:

$$x^i_{k,l} = \max(|z^i_{k,l}| - \frac{1}{\beta}, 0) \frac{z^i_{k,l}}{|z^i_{k,l}|} \tag{8}$$

where all operations are element-wise. Alternatively for arbitrary values of $p > 0$, the optimal solution can be computed via a lookup-table [11]. This permits us to impose more aggressive forms of sparsity than $p = 1$.

Filter updates : With x fixed and $z^i_{k,l}$ computed for a fixed i, we use the following for gradient updates of $f^l_{k,c}$:

$$\frac{\partial \hat{C}_l(y)}{\partial f^l_{k,c}} = \lambda \sum_{i=1}^{I} \sum_{\tilde{c}=1}^{K_{l-1}} Z^{i^T}_{k,l} (\sum_{\tilde{k}=1}^{K_l} g^l_{k,c} Z^i_{\tilde{k},l} f^l_{\tilde{k},\tilde{c}} - z^i_{\tilde{c},l-1}) \tag{9}$$

where Z is a convolution matrix similar to F. The overall learning procedure is summarized in Algorithm 1.

Algorithm 1 : Learning a single layer, l, of the Deconvolutional Network.

Require: Training images y, # feature maps K, connectivity g
Require: Regularization weight λ, # epochs E
Require: Continuation parameters: $\beta_0, \beta_{\text{Inc}}, \beta_{\text{Max}}$
Initialize feature maps and filters $z \sim \mathcal{N}(0, \epsilon)$, $f \sim \mathcal{N}(0, \epsilon)$
for epoch $= 1 : E$ **do**
  for $i = 1 : I$ **do**
    $\beta = \beta_0$
    while $\beta < \beta_{\text{Max}}$ **do**
      Given $z^i_{k,l}$, solve for $x^i_{k,l}$ using Eqn. 8, $\forall k$.
      Given $x^i_{k,l}$, solve for $z^i_{k,l}$ using Eqn. 6, $\forall k$.
      $\beta = \beta \cdot \beta_{\text{Inc}}$
    end while
  end for
  Update $f^l_{k,c}$ using gradient descent on Eqn. 9, $\forall k, c$.
end for
Output: Filters f

3.3. Image representation/reconstruction

To use the model for image reconstruction, we first decompose an input image by using the learned filters f to find the latent representation z. We explain the procedure for a 2 layer model. We first infer the feature maps $z_{k,1}$ for layer 1 using the input y' and the filters $f^1_{k,c}$ by minimizing $C_1(y')$. Next we update the feature maps for layer 2, $z_{k,2}$ in an alternating fashion. In step 1, we first minimize the reconstruction error w.r.t. y', projecting $z_{k,2}$ through $f^2_{k,c}$ and $f^1_{k,c}$ to the image:

$$\frac{\lambda_2}{2} \sum_{c=1}^{K_0} \| \sum_{k=1}^{K_1} g^1_{k,c} (\sum_{b=1}^{K_2} g^2_{b,k} (z_{b,2} \oplus f^2_{b,k})) \oplus f^1_{k,c}) - y'_c \|_2^2 + \sum_{k=1}^{K_2} |z_{k,2}| \tag{10}$$

[3] $F^l_{k,c} z^i_{k,l} \equiv z^i_{k,l} \oplus f^l_{k,c}$ and $F^{l^T}_{k,c} z^i_{k,l} \equiv z^i_{k,l} \oplus$ `flipud(fliplr(` $f^l_{k,c}$ `))` using Matlab notation.

In step 2, we minimize the error w.r.t. $z_{k,2}$:

$$\frac{\lambda_2}{2}\sum_{c=1}^{K_1}\|\sum_{k=1}^{K_2} g_{k,c}^2(z_{k,2}\oplus f_{k,c}^2) - z_{c,1}\|_2^2 + \sum_{k=1}^{K_2}|z_{k,2}| \quad (11)$$

We alternate between steps 1 and 2, using conjugate gradient descent in both. Once $z_{k,2}$ has converged, we reconstruct y' by projecting back to the image via $f_{k,c}^2$ and $f_{k,c}^1$:

$$\tilde{y}' = \sum_{k=1}^{K_1} g_{k,c}^1(\sum_{b=1}^{K_2} g_{b,k}^2(z_{b,2}\oplus f_{b,k}^2))\oplus f_{k,c}^1 \quad (12)$$

An important detail is the addition of an extra feature map z_0 per input map of layer 1 that connects to the image via a constant uniform filter f_0. Unlike the sparsity priors on the other feature maps, z_0 has an ℓ_2 prior on the gradients of z_0, i.e. the prior is of the form $\|\nabla z_0\|^2$. These maps capture the low-frequency components, leaving the high-frequency edge structure to be modeled by the learned filters. Given that the filters were learned on high-pass filtered images, the z_0 maps assist in reconstructing raw images.

4. Experiments

In our experiments, we train on two datasets of 100×100 images, one containing natural scenes of fruits and vegetables and the other consisting of scenes of urban environments. In all our experiments, unless otherwise stated, the same learning settings were used for all layers, namely: $H=7$, $\lambda=1$, $p=1$, $\beta_0=1$, $\beta_{\text{Inc}}=6$, $\beta_{\text{Max}}=10^5$, $E=3$.

4.1. Learning multi-layer deconvolutional filters

With the settings described above we trained a separate 3 layer model for each dataset, using an identical architecture. The first layer had 9 feature maps fully-connected to the input. The second layer had 45 maps: 36 were connected to pairs of maps in the first layer, and the remainder were singly-connected. The third layer had 150 feature maps, each of which was connected to a random pair of second layer feature maps. In Fig. 7 and Fig. 8 we show the filters that spontaneously emerge, projected back into pixel space.

The first layer in each model learns Gabor-style filters, although for the city images they are not evenly distributed in orientation, preferring vertical and horizontal structures. The second layer filters comprise an assorted set of V2-like elements, with center-surround, corners, T-junctions, angle-junctions and curves. The third layer filters are highly diverse. Those from the model trained on food images (Fig. 7) comprise several types: oriented gratings (rows 1–4); blobs (D8, E7, H9); box-like structures (B10, F12) and others that capture parallel and converging lines (C12, J11). The filters trained on city images (Fig. 8) capture line groupings in horizontal and vertical configurations. These include: conjunctions of T-junctions (C15, G11); boxes (D14, E4) and various parallel lines (B15, D8, I3). Some of the filters are representative of the tokens shown in Fig. 2-4 of Marr [18] (see Fig. 1).

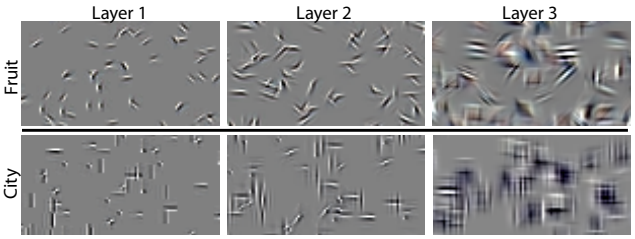

Figure 3. Samples from the layers of two deconvolutional network models, trained on fruit (top) or city (bottom) images.

Since our model is generative, we can sample from it. In Fig. 3 we show samples from the two different models from each level projected down into pixel space. The samples were drawn using the relative firing frequencies of each feature from the training set.

4.2. Comparison to patch-based decomposition

To demonstrate the benefits of imposing sparsity within a convolutional architecture, we compare our model to the patch-based sparse decomposition approach of Mairal *et al.* [16]. Using the SPAMS code accompanying [16] we performed a patch-based decomposition of the two image sets, using 100 dictionary elements. The resulting filters are shown in Fig. 4(left). We then attempted to build a hierarchical 2 layer model by taking the sparse output vectors from each image patch and arranging them into a map over the image. Applying the SPAMS code to this map produces the 2nd layer filters shown in Fig. 4(right). While larger in scale than the 1st layer filters, they are generally Gabor-like and do not show the diverse edge conjunctions present in our 2nd layer filters. To probe this result, we visualize the latent feature maps of our convolutional decomposition and Mairal et. al.'s patch-based decomposition in Fig. 5.

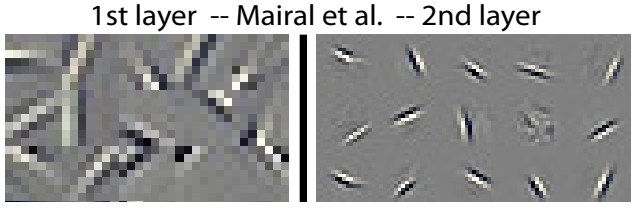

Figure 4. Examples of 1st and 2nd layer filters learned using the patch-based sparse deconvolution approach of Mairal *et al.* [16], applied to the food dataset. While the first layer filters look similar to ours, the 2nd layer filters are merely larger versions of the 1st layer filters, lacking the edge compositions found in our 2nd layer (see Fig. 7 and Fig. 8).

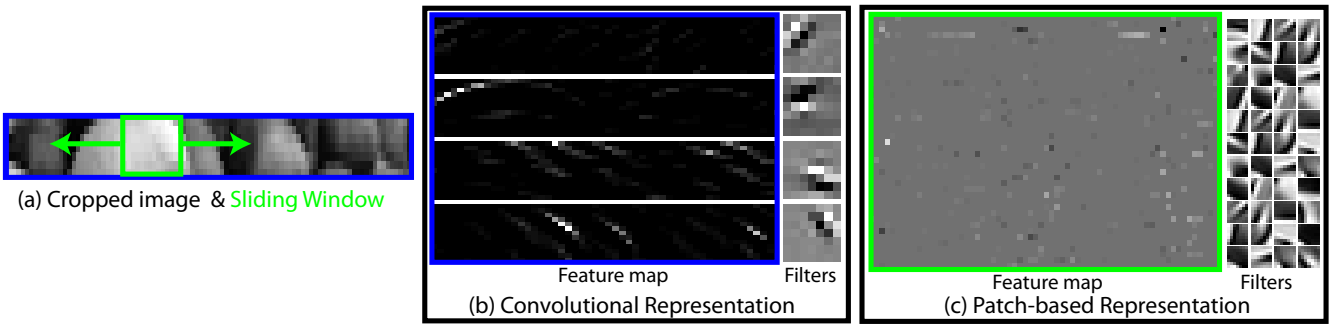

Figure 5. A comparison of convolutional and patch-based sparse representations for a crop from a natural image (a). (b): Sparse convolutional decomposition of (a). Note the smoothly varying feature maps that preserve spatial locality. (c): Patch-based convolutional decomposition of (a) using a sliding window (green). Each column in the feature map corresponds to the sparse vector over the filters for a given x-location of the sliding window. As the sliding window moves the latent representation is highly unstable, changing rapidly across edges. Without a stable representation, stacking the layers will not yield higher-order filters, as demonstrated in Fig. 4.

Table 1. Recognition performance on Caltech-101.

# training examples	15	30
DN-1 (KM)	$57.7 \pm 1.0\%$	$65.8 \pm 1.3\%$
DN-2 (KM)	$57.0 \pm 0.8\%$	$65.5 \pm 1.0\%$
DN-(1+2) (KM)	**58.6** $\pm 0.7\%$	**66.9** $\pm 1.1\%$
Lazebnik *et al.* [12]	56.4%	$64.6 \pm 0.7\%$
Jarret *et al.* [9]	–	$65.6 \pm 1.0\%$
Lee *et al.* [15] layer-1	$53.2 \pm 1.2\%$	$60.5 \pm 1.1\%$
Lee *et al.* [15] layer-1+2	$57.7 \pm 1.5\%$	$65.4 \pm 0.5\%$
Zhang *et al.* [29]	$59.1 \pm 0.6\%$	$66.2 \pm 0.5\%$

4.3. Caltech-101 object recognition

We now demonstrate how Deconvolutional Networks can be used in an object recognition setting. As we are primarily interested in image representation, we compare to other methods using a common framework of one or more layers of feature extraction, followed by Spatial Pyramid Matching [12]. We use the standard Caltech-101 dataset for evaluating classification performance, but we would like to emphasize that the filters of our DN have been learned using a generic, disparate training set: a concatenation of the natural and city images. The Caltech-101 images are only used for supervised training[4] of the classifier.

Our baseline is the method of Lazebnik *et al.* [12] where SIFT descriptors are computed densely over the image, followed by Spatial Pyramid Matching. To compare our latent representation with this approach, we densely constructed descriptors[5] from layer 1 (DN-1) and layer 2 (DN-2) feature activations. These were then vector quantized using K-means (KM) into 1000 clusters and grouped into a spatial pyramid from which an SVM histogram intersection kernel was computed for classification. Results for 10-fold cross validation with 15 and 30 images training per category are reported in Table 1.

Our method slightly outperforms the SIFT-based approach [12] as well as other multi-stage convolutional feature-learning methods such as convolutional DBNs [15] and feed-forward convolutional networks [9]. We achieved the best performance when we concatenated the spatial pyramids of both layers before computing the SVM histogram intersection kernels: denoted DN-(1+2).

4.4. Denoising images

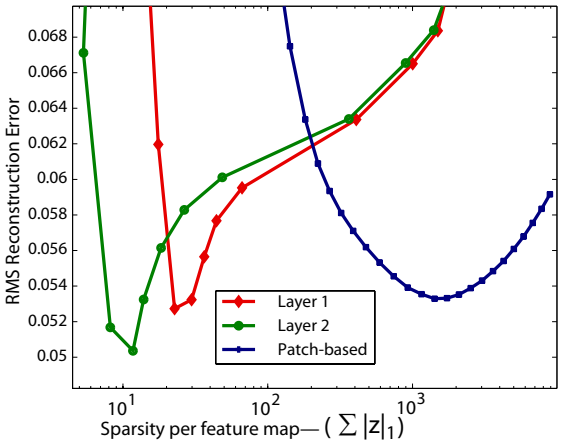

Figure 6. Exploring the trade-off between sparsity and denoising performance for our 1st and 2nd layer representations (red and green respectively), as well as the patch-based approach of Mairal *et al.* [16] (blue). Our 2nd layer representation simultaneously achieves a lower reconstruction error and sparser feature maps.

Given that our learned representation can be used for synthesis as well as analysis, we explore the ability of a two

[4]The 150x150 pixel contrast normalized gray images used for classification were connected to 8 feature maps in layer 2. Second layer maps were connected singly and in every possible pair to the layer 1 maps, for a total of 36 layer 2 feature maps. p=0.8, λ_1=10, and λ_2=1 were used to maintain more discriminative information in the feature maps.

[5]Activations from each layer were split into overlapping 16x16 patches at a stride of 2 pixels. The absolute value of activations in each patch were pooled by a factor of 4 then grouped in 4x4 regions on each of 8 layer 1 feature maps giving a 128-D descriptor per patch and grouped in 2x2 regions on each of 36 layer 2 maps leading to 144-D layer 2 descriptors.

layer model to denoise images. Applying Gaussian noise to an image with a SNR of 13.84dB, the first layer of our model was able to reduce the noise to 16.31dB. Further, using the latent features of our second layer to reconstruct the image, the noise was reduced to a SNR of 18.01dB.

We also explore the relative sparsity of the feature maps in the 1^{st} and 2^{nd} layers of our model as we vary λ. In Fig. 6 we plot the average sparsity of each feature map against RMS reconstruction error, we see that the feature maps at layer 2 are sparser and give a lower reconstruction error, than those of layer 1. We also plot the same curve for the patch-based sparse decomposition of Mairal *et al.* [16]. In this framework, inference is performed separately for each image patch and since patches overlap, a much larger number of latent features are needed to represent the image. The curve was produced by varying the number of active dictionary atoms per patch in reconstruction.

4.5. Inference timings

Our efficient optimization scheme makes it feasible to perform exact inference in a convolutional setting. Alternate approaches [15] rely on simple non-linear encoders to perform approximate inference. Our scheme is linear in the number of filters and pixels in the image (5.8 ± 1.0secs/filter/megapixel) Thus for 150×150 images used in the Caltech 101 experiments, using the architecture described in Section 4.1, inferences takes 2.5s, 10s, 55s layers-1,2,3 respectively. Due to the small filter sizes, learning incurs only a 10% overhead relative to inference. While our algorithm is slow compared to approaches that use bottom-up encoders, heavy use of the convolution operator makes it amenable to parallelization and GPU-based implementations which we expect would give between 1 and 2 orders of magnitude speed-up. Additional performance gains could result from introducing pooling between layers.

5. Conclusion

We have introduced Deconvolutional Networks: a conceptually simple framework for learning sparse, over-complete feature hierarchies. Applying this framework to natural images produces a highly diverse set of filters that capture high-order image structure beyond edge primitives. These arise without the need for hyper-parameter tuning or additional modules, such as local contrast normalization, max-pooling and rectification [9]. Our approach relies on robust optimization techniques to minimize the poorly conditioned cost functions that arise in the convolutional setting. Supplemental images, video, and code can be found at: `http://www.cs.nyu.edu/~zeiler/pubs/cvpr2010/`.

References

[1] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11(7):1691–1715, 1999.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, pages 153–160, 2007.

[3] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci Comp.*, 20(1):33–61, 1999.

[4] S. Fidler, M. Boben, and A. Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. In *CVPR*, 2008.

[5] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*, 2007.

[6] D. Geman and Y. C. Nonlinear image recovery with half-quadratic regularization. *PAMI*, 4:932–946, 1995.

[7] C. E. Guo, S. C. Zhu, and Y. N. Wu. Primal sketch: Integrating texture and structure. *CVIU*, 106:5–19, April 2007.

[8] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

[9] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[10] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, volume 2, pages 2145–2152, 2006.

[11] D. Krishnan and R. Fergus. Analytic Hyper-Laplacian Priors for Fast Image Deconvolution. In *NIPS*, 2009.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–24, 1998.

[14] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2007.

[15] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009.

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, pages 689–696, 2009.

[17] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS*, 2008.

[18] D. Marr. *Vision*. Freeman, San Francisco, 1982.

[19] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[20] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007.

[21] M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *NIPS*. MIT Press, 2008.

[22] M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS*, pages 1137–1144, 2006.

[23] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[24] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005.

[25] Z. W. Tu and S. C. Zhu. Parsing images into regions, curves, and curve groups. *IJCV*, 69(2):223–249, August 2006.

[26] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[27] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM J. Imag. Sci.*, 1(3):248–272, 2008.

[28] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[29] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

[30] L. Zhu, Y. Chen, and A. L. Yuille. Learning a hierarchical deformable template for rapid deformable object parsing. *PAMI*, March 2009.

[31] S. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Comp. Graphics and Vision*, 2(4):259–362, 2006.

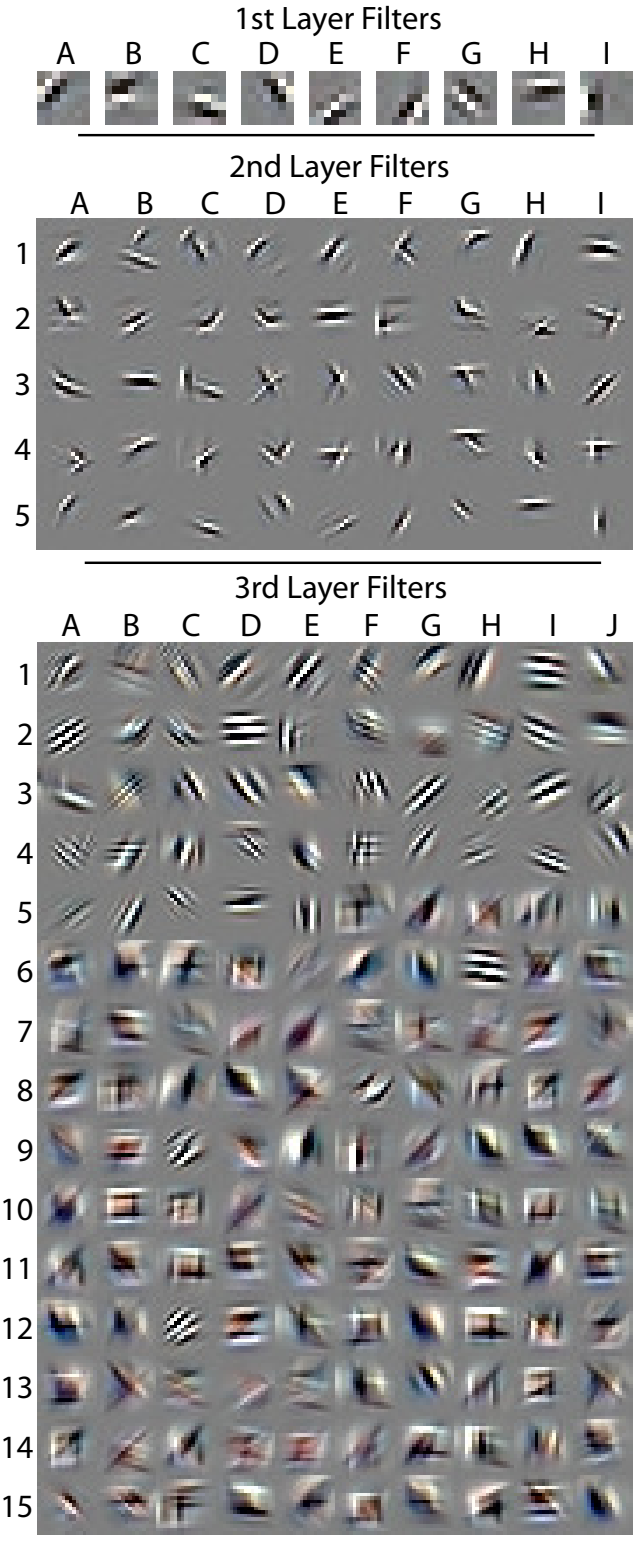

Figure 7. Filters from each layer in our model, trained on food scenes. Note the rich diversity of filters and their increasing complexity with each layer. In contrast to the filters shown in Fig. 8, the filters are evenly distributed over orientation.

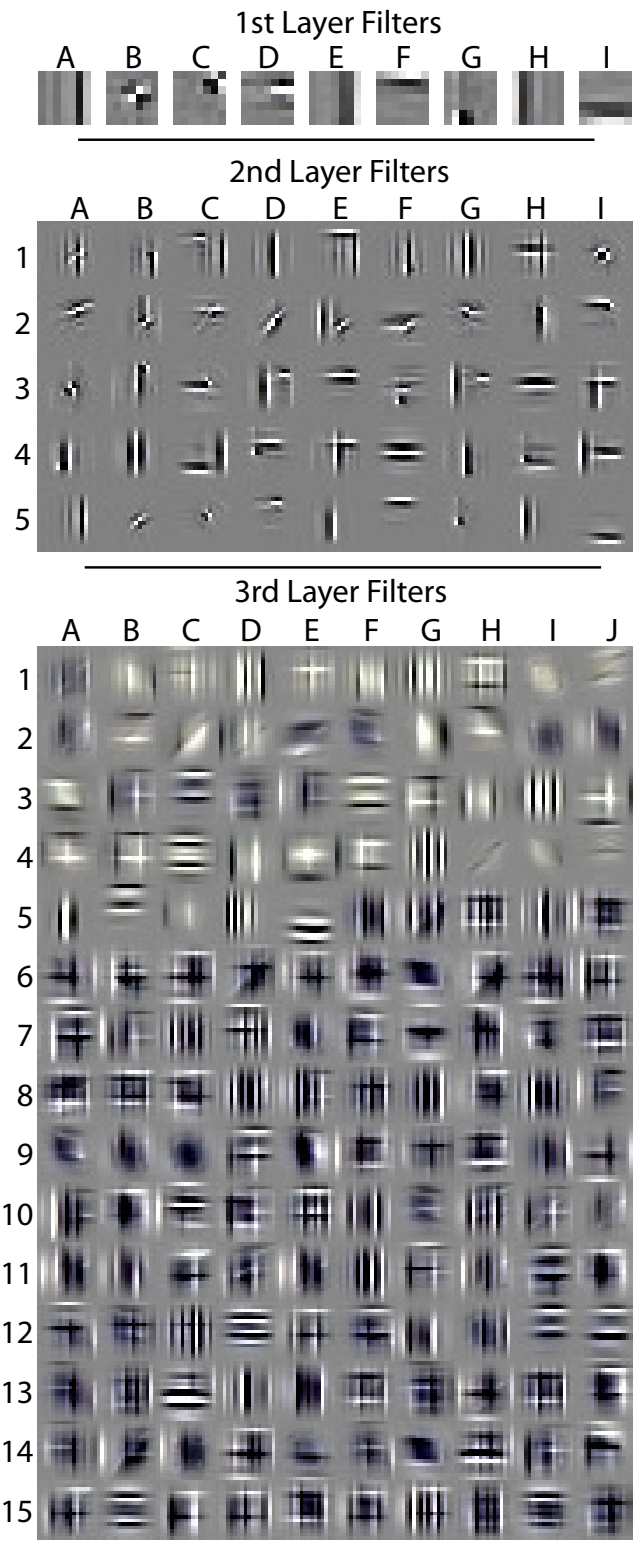

Figure 8. Filters from each layer in our model, trained on the city dataset. Note the predominance of horizontal and vertical structures.