Unsupervised Learning of Compositional Sparse Code for Natural Image Representation

Yi Hong, Zhangzhang Si, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu
Departments of Statistics and Computer Science
University of California, Los Angeles

Abstract

This article proposes an unsupervised method for learning compositional sparse code for representing natural images. Our method is built upon the original sparse coding framework where there is a dictionary of basis functions often in the form of localized, elongated and oriented wavelets, so that each image can be represented by a linear combination of a small number of basis functions automatically selected from the dictionary. In our compositional sparse code, the representational units are composite: they are compositional patterns formed by the basis functions. These compositional patterns can be considered shape templates. We propose an unsupervised learning method for learning a dictionary of frequently occurring templates from training images, so that each training image can be represented by a small number of templates automatically selected from the learned dictionary. The compositional sparse code translates the raw image of a large number of pixel intensities into a small number of templates, thus facilitating the signal to symbol transition and allowing a symbolic description of the image. Experiments show that our method is capable of learning meaningful compositional sparse code, and the learned templates are useful for image classification.

Keywords: Compositionality, Sparse coding, Structured sparsity.

1 Introduction

1.1 Motivation and objectives

As illustrated by Figure 1, the ancient Chinese developed the early form of Chinese characters as a coding scheme for representing natural images where each character is a pictorial description of a pattern. The early pictorial form then gradually evolved into the form that is in use today. The system of Chinese characters can be considered a compositional sparse code: each natural image can be described by a small number of characters selected from the dictionary, and each character is a composition of a small number of strokes.

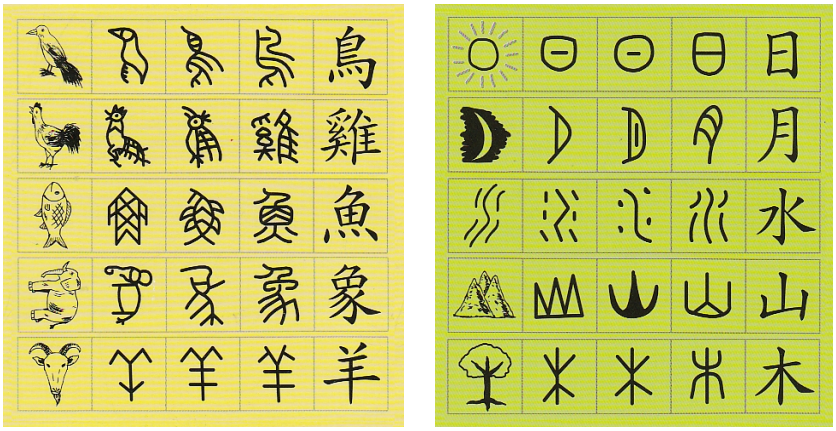
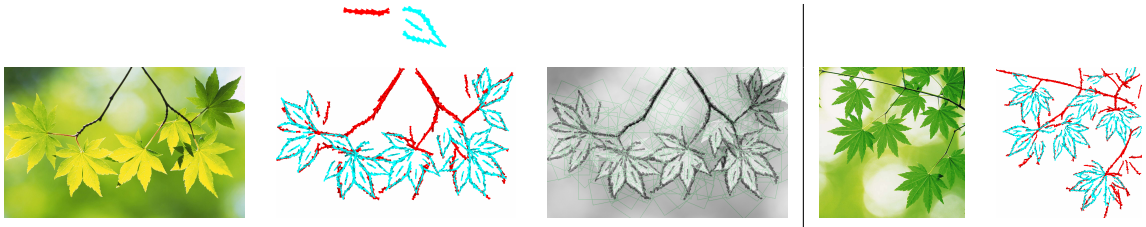


Figure 1: Chinese characters evolved from representations of natural images of objects and scenes [18]. In each row, the first block shows a picture of the object, and the rest four blocks display the evolution of the corresponding Chinese character over time. Left panel: bird, chicken, fish, elephant and goat. Right panel: sun, moon, water, mountain and wood.

The goal of this paper is to develop a compositional sparse code for natural images. Our coding scheme can be viewed as a mathematical realization of the system of the Chinese characters. In our compositional sparse code, each "stroke" is a linear basis function such as a Gabor wavelet, and each "character" is a compositional pattern or a shape template formed by a small number of basis functions. We propose an unsupervised learning method for learning the frequently occurring templates from training images, so that each training image can be represented by a small number of templates automatically selected from the learned dictionary. Such a compositional sparse code translates the original raw image of a large number of pixel intensities into a small number of templates, thus facilitating the signal to symbol transition and allowing a symbolic description of the image. Our experiments show that our method is capable of learning meaningful compositional sparse code. Experiments also show that the learned templates can be useful for image classification.

For example, it enables us to learn meaningful "words" in the so-called "bag-of-words" classification scheme.



(a) (b) (c) (d) (e)

Figure 2: Unsupervised learning of compositional sparse code (a,b,c) and using it for recognition and segmentation (d,e). (a) Training image of 480 $\times$ 768 pixels. (b) Above: 2 compositional patterns (twig and leaf) in the form of shape templates learned from the training image. Below: Representing the training image by translated, rotated, scaled and deformed copies of the 2 templates. (c) Superposing the deformed templates on the original image. Green squared boxes are bounding boxes of the templates. (d) Testing image. (e) Representation (recognition) of the testing image by the 2 templates.

Figure 2 illustrates the basic idea. We start with a dictionary of Gabor wavelets centered at a dense collection of locations and tuned to a collection of scales and orientations. In Figure 2, each Gabor wavelet is illustrated by a bar at the same location and with the same length and orientation as the corresponding wavelet. Figure 2.(a) displays the training image. (b) displays a mini-dictionary of 2 compositional patterns of wavelets learned from the training image. Each compositional pattern is a template formed by a group of a small number of wavelets at selected locations and orientations. The learning is unsupervised in the sense that the images are not labeled or annotated. The number of templates in the dictionary is automatically determined by an adjusted BIC criterion. The 2 templates are displayed in different colors, so that it can be seen clearly how the translated, rotated, scaled and deformed copies of the 2 templates are used to represent the training image, as shown in (b). In (c), the templates are overlaid on the original image, where each green squared box is the bounding box of the template. In our current implementation, we allow some overlap between the bounding boxes of the templates. The templates learned from the training image can be generalized to testing images, as shown in (d) and (e).

Figure 3 shows another example, where part templates of egrets and templates of water waves and grasses are learned from 20 training images without supervision. It is interesting to observe that in this example, the unsupervised learning also accomplishes image segmentation, object detection and perceptual grouping (e.g., grass pattern), which are important tasks in vision.

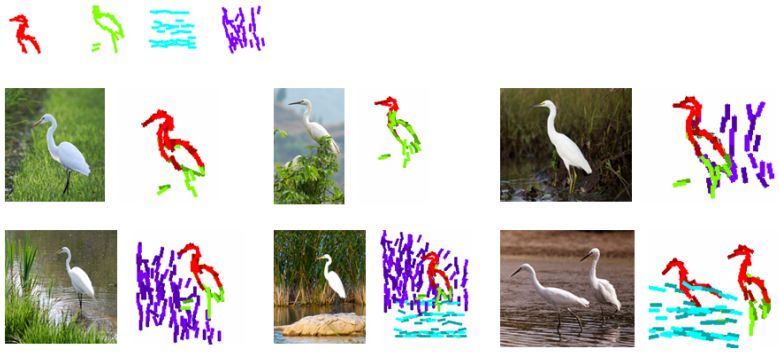


Figure 3: Another example. Four compositional patterns (templates) are learned unsupervisedly from 20 training images. The bounding box of the templates is 100×100. The numbers of basis functions (Gabor wavelets in our experiments) are less than 40, and the actual numbers are automatically determined.

Our compositional sparse code combines two fundamental principles in image representation and computational vision, namely, *sparsity* and *compositionality*. We shall briefly review these two principles below and then give an overview of our methodology.

1.2 Sparsity and its limitation

Recent years have seen a flurry of research activities on sparsity in applied mathematics, statistics and machine learning. Sparsity also plays a fundamental role in representing natural images which are functions defined on a two-dimensional domain (such as a lattice). According to the sparsity principle [24]), there is a dictionary of basis functions defined on the same image domain, so that each natural image can be represented by a linear superposition of a small number of basis functions automatically selected from the dictionary. By enforcing sparsity, Olshausen and Field [24] were able to learn a dictionary of basis functions from natural image patches, and these basis functions resemble Gabor wavelets at different locations, orientations and scales. The Gabor wavelets are considered a mathematical model for the so-called simple cells in the primary visual cortex or V1 [5]. Olshausen and Field proposed that the role of V1 is to infer sparse representations of natural images.

The dictionary learned by Olshausen and Field is over-complete, meaning that the number of basis functions in the dictionary is greater than the number of pixels in the image domain. An advantage of such an over-complete dictionary is that the basis functions in this rich dictionary can afford to be specific enough so that each image can be represented by only a small number of basis functions selected from the dictionary. A popular method for selecting the basis functions from a given dictionary is matching pursuit [20], which is a greedy algorithm that selects one basis

function in each iteration, which seeks the maximal reduction in the least squares reconstruction error. A related method is basis pursuit [3] or Lasso [29], which selects the basis functions by solving a penalized least squares problem where the penalty is in the form of the ℓ_1 norm of the coefficients of the basis functions.

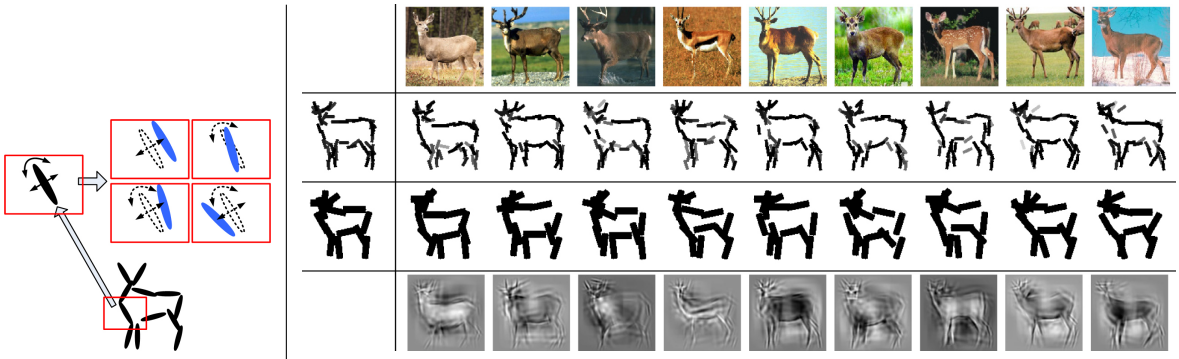
In the sparse coding framework, the basis functions in the dictionary exist individually without any structures imposed on them, and the coefficients of these basis functions are usually assumed to be independent of each other. Such simplified independence assumption is clearly inadequate for modeling the wide varieties of patterns in natural images. It is necessary to discover patterns and structures in the selected basis functions.

1.3 Compositionality and structured sparsity

The compositionality principle was proposed in the context of vision by Geman, Potter, and Chi [14] and Zhu and Mumford [37]. The principle holds that patterns in natural images are compositions of parts, which are themselves compositions of sub-parts, and so on. An interesting example cited by Geman et al. is Laplace's remark that one strongly prefers to view the string CONSTANTINOPLE as a single word, rather than 14 individual letters. This is also the case with the basis functions in the sparse coding of natural images. Like letters forming the words, the basis functions in the sparse representations of natural images also form various compositional patterns in terms of their spatial arrangements. We call such sparsity the compositional sparsity, which is a special form of structured sparsity.

Structured sparsity has received considerable attention in statistics and machine learning in recent years. The most prominent example is the group Lasso [33], which replaces the ℓ_1 penalty of Lasso by a composite penalty based on the group structure among the basis functions. In the group Lasso, the collection of the groups is assumed given. In our work, however, we do not assume that the groups are given, and we seek to learn dictionaries of the recurring compositional patterns in the spatial grouping of the basis functions.

Any hierarchical compositional model will necessarily end with constituent elements that cannot be further decomposed, and such elements may be called "atoms." Interestingly, the basis functions are commonly referred to as atoms in sparse coding literature, and the sparse representation based on atoms is usually called "atomic decomposition" [7]. Compositionality enables us to compose atoms into composite representational units, which lead to much sparser and thus more meaningful representations of the signals.



(a) (b)

Figure 4: (a) An active basis model is a composition of a small number of basis functions, each is a Gabor wavelet and is illustrated by a bar with the same location, orientation and length. Each basis function can perturb its location and orientation. (b) Supervised learning of active basis model from aligned images. The first row displays the 9 training images. The second row: the first plot is the nominal template formed by 50 basis functions. The rest of the plots are the deformed templates matching the images respectively. The third row: the same as the second row, except that the scale of the Gabor wavelets is about twice as large, and the number of wavelets is 14. The last row displays the linear reconstruction of each training image from 100 selected and perturbed basis functions.

1.4 Overview of our methodology

In this article, we assume that the dictionary of basis functions is a given, and they are Gabor wavelets at a dense collection of locations, orientations and scales. We focus on learning compositional patterns formed by these basis functions. In principle, the basis functions can also be learned from training images, and we are investigating this matter in our on-going work.

We represent each compositional pattern of basis functions by an active basis model of Wu et al. [32], which is a composition of a small number of basis functions automatically selected from a given dictionary. The selected basis functions are allowed to perturb their locations and orientations so that the linear basis formed by the selected basis functions become active and the active basis can be viewed a deformable template. Figure 4 illustrates the basic idea of the active basis model.

Wu et al. mainly studied the problem of learning a single active basis model from a set of aligned training images, where the images are defined on the same bounding box, and the objects in the images are from the same category, in the same pose, and appear at the same location and scale. The learning of the active basis model in this situation can be called supervised learning.

Our work goes far beyond Wu et al. In our work, the training images are not assumed to be aligned, and each image can be represented by multiple active basis templates. That is, the active

basis models serve as the composite representational units in our compositional sparse code. For a given set of training images, our method is to learn a dictionary of active basis templates, so that each training image can be represented by a small number of templates that are translated, rotated, scaled and deformed copies of the learned templates in the dictionary.

The unsupervised learning algorithm starts from templates learned from randomly cropped image patches, so the initial templates are rather random. The algorithm then iterates the following two steps:

1. Image encoding: Encode each training image by translated, rotated, scaled and deformed copies of the templates in the current dictionary, by a template matching pursuit process.

2. Dictionary learning: Re-learn each template from image patches currently encoded by this template, by a shared matching pursuit process.

The rest of the paper is organized as follows. Section 2 reviews the original sparse coding framework and the active basis model. Section 3 presents our representational scheme and the unsupervised learning algorithm. Section 4 presents experimental results on image representation and classification. Section 5 concludes with a discussion.

2 Background: sparse coding model and active basis model

This section reviews sparse coding model and the active basis model in order to fix the notation and set the stage for presenting our model and learning algorithm in the next section.

2.1 Olshausen-Field model for sparse coding

Olshausen and Field [24] proposed that the role of simple V1 cells is to compute sparse representations of natural images. Let $\{\mathbf{I}_m, m = 1, ..., M\}$ be a set of training image patches (e.g. 12×12), which are two-dimensional functions defined on a certain image domain. The Olshausen-Field model seeks to represent these images by

$$\mathbf{I}_m = \sum_{i=1}^{N} c_{m,i} B_i + U_m, \tag{1}$$

where $(B_i, i = 1, ..., N)$ is a dictionary of basis functions defined on the same domain as $\mathbf{I}_m$, $c_{m,i}$ are the coefficients, and U_m is the unexplained residual image. N is often assumed to be greater than the number of pixels in $\mathbf{I}_m$ (e.g. $N = 2 \times 12 \times 12$), so the dictionary is said to be over-complete. On the other hand, the number of coefficients $(c_{m,i}, i = 1, ..., N)$ that are non-zero (or significantly different from zero) is assumed to be small (e.g., less than 10) for each image $\mathbf{I}_m$.

Geometric attributes. One may also assume that the basis functions in the dictionary are translated, rotated and dilated versions of one another, as in [25], so that each B_i can be written as $B_{x,s,\alpha}$, where x is the location (a two-dimensional vector), s is the scale, and α is the orientation. We call such a dictionary self-similar, and we call (x, s, α) the geometric attribute of $B_{x,s,\alpha}$.

Model (1) then becomes

$$\mathbf{I}_m = \sum_{x,s,\alpha} c_{m,x,s,\alpha} B_{x,s,\alpha} + U_m, \tag{2}$$

where $B_{x,s,\alpha}$ are translated, rotated and dilated copies of a single basis function, e.g., $B = B_{x=0,s=1,\alpha=0}$, and (x, s, α) are properly discretized (default setting: α is discretized into 16 equally spaced orientations). B can be learned from training images $\{\mathbf{I}_m\}$ [25].

Assumption on basis functions in this paper. From now on, we assume that the dictionary of basis functions is self-similar, and $\{B_{x,s,\alpha}, \forall(x, s, \alpha)\}$ is already given. It can either be learned or designed. In the following, we assume that $B_{x,s,\alpha}$ is a Gabor wavelet, and we also assume that $B_{x,s,\alpha}$ is normalized to have unit ℓ_2 norm so that $|B_{x,y,\alpha}|^2 = 1$. $B_{x,s,\alpha}$ may also be a pair of Gabor sine and cosine wavelets, so that for each Gabor wavelet B, $B = (B_0, B_1)$. The corresponding coefficient $c = (c_0, c_1)$, and $cB = c_0 B_0 + c_1 B_1$. The projection $\langle \mathbf{I}, B\rangle = (\langle \mathbf{I}, B_0\rangle, \langle \mathbf{I}, B_1\rangle)$, and $|\langle \mathbf{I}, B\rangle|^2 = \langle \mathbf{I}, B_0\rangle^2 + \langle \mathbf{I}, B_1\rangle^2$.

Spatial point process. Given the dictionary $(B_{x,s,\alpha}, \forall(x, s, \alpha))$, the encoding of an image $\mathbf{I}_m$ amounts to inferring $(c_{m,x,s,\alpha}, \forall(x, s, \alpha))$ in (2) under the sparsity constraint, which means that only a small number of $(c_{m,x,s,\alpha})$ are non-zero. That is, we seek to encode $\mathbf{I}_m$ by

$$\mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{x_{m,i}, s_{m,i}, \alpha_{m,i}} + U_m, \tag{3}$$

where $n \ll N$ is a small number, and $(x_{m,i}, s_{m,i}, \alpha_{m,i}, i = 1, ..., n)$ are the geometric attributes of the selected basis functions whose coefficients $(c_{m,i})$ are non-zero. $(x_{m,i}, s_{m,i}, \alpha_{m,i}, i = 1, ..., n)$ form a spatial point process (we continue to use i to index the basis functions, but here i only runs through the n selected basis functions instead of all the N basis functions as in (1)).

2.2 Active basis model for shared sparse coding of aligned image patches

The active basis model was proposed by Wu et al. [32] for modeling deformable templates formed by basis functions.

Suppose we have a set of training image patches $\{\mathbf{I}_m, m = 1, ..., M\}$. This time we assume that they are defined on the same bounding box, and the objects in these images come from the same category. In addition, these objects appear at the same location, scale and orientation, and in the same pose. See Figure 4 for 9 image patches of deer. We call such image patches aligned.

The active basis model is of the following form

$$\mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}} + U_m, \tag{4}$$

where $\mathbf{B} = (B_{x_i,s,\alpha_i}, i = 1, ..., n)$ form the nominal template of an active basis model (sometimes we simply call $\mathbf{B}$ an active basis template). Here we assume that the scale s is fixed and given. $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$ is the deformed version of the nominal template $\mathbf{B}$ for encoding $\mathbf{I}_m$, where $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ are the perturbations of the location and orientation from the nominal location x_i and the nominal orientation α_i respectively. The perturbations are introduced to account for shape deformation. Both $\Delta x_{m,i}$ and $\Delta\alpha_{m,i}$ are assumed to vary within limited ranges (default setting: $\Delta x_{m,i} \in [-3, 3]$ pixels, and $\Delta\alpha_{m,i} \in \{-1, 0, 1\} \times \pi/16$).

2.3 Prototype algorithm

Given the dictionary of basis functions $\{B_{x,s,\alpha}, \forall x, s, \alpha\}$, the learning of the active basis model from the aligned image patches $\{\mathbf{I}_m\}$ involves the sequential selection of B_{x_i,s,α_i} and the inference of its perturbed version $B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}$ in each image $\mathbf{I}_m$. We call the learning as supervised, because the bounding boxes of the objects are given and the images are aligned. See Figure 4 for an illustration of the learning results.

In this subsection, we consider a prototype version of the shared matching pursuit algorithm, which is to be revised in the following subsections. The reason we start from this prototype algorithm is that it is simple and yet captures the key features of the learning algorithm.

The prototype algorithm is a greedy algorithm that seeks the maximal reduction of the following least squares reconstruction error

$$\sum_{m=1}^{M} |\mathbf{I}_m - \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}|^2 \tag{5}$$

in each iteration (recall that the basis functions are normalized to have unit ℓ_2 norm).

[0] Initialize $i \leftarrow 0$. For $m = 1, ..., M$, initialize the residual image $U_m \leftarrow \mathbf{I}_m$.

[1] $i \leftarrow i + 1$. Select the next basis function by

$$(x_i, \alpha_i) = \arg\max_{x,\alpha} \sum_{m=1}^{M} \max_{\Delta x,\Delta\alpha} |\langle U_m, B_{x+\Delta x,s,\alpha+\Delta\alpha}\rangle|^2,$$

where $\max_{\Delta x,\Delta\alpha}$ is a local maximum pooling within the small ranges of $\Delta x_{m,i}$ and $\Delta\alpha_{m,i}$.

[2] For $m = 1, ..., M$, given (x_i, α_i), infer the perturbations in location and orientation by retrieving the arg-max in the local maximum pooling of step [1]:

$$(\Delta x_{m,i}, \Delta\alpha_{m,i}) = \arg\max_{\Delta x,\Delta\alpha} |\langle U_m, B_{x_i+\Delta x,s,\alpha_i+\Delta\alpha}\rangle|^2.$$

Let $c_{m,i} \leftarrow \langle U_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle$, and update the residual image by explaining away:

$$U_m \leftarrow U_m - c_{m,i} B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}. \tag{6}$$

[3] Stop if $i = n$, else go back to step [1].

Assumption on orthogonality in this paper. Because of the arg-max explaining-away (6), the basis functions selected for each deformed template $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$ usually have little overlap with each other. For computational and modeling convenience, we shall assume that these selected basis functions are orthogonal to each other, so that the coefficient can be obtained by projection: $c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle$. We write $C_m = (c_{m,i}, i = 1, ..., n)$. In practice, we allow small overlap between the selected basis functions in each $\mathbf{B}_m$.

2.4 Statistical modeling

Density substitution. The above algorithm guided by (5) implicitly assumes that the residual U_m is Gaussian white noise. This assumption can be problematic because the unexplained background in the image may contain salient structures such as edges. This is why we need to revise the above algorithm which is based on the Gaussian white noise assumption. A better assumption is to assume that U_m follows the same distribution as that of natural images. More precisely, the distribution of $\mathbf{I}_m$ given the deformed template $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$, i.e., $p(\mathbf{I}_m \mid \mathbf{B}_m)$, is obtained by modifying the distribution of natural images $q(\mathbf{I}_m)$ in such a way that we only change the distribution of $C_m = (c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle, i = 1, ..., n)$ from $q(C_m)$ to $p(C_m)$, while leaving the conditional distribution of U_m given C_m unchanged. Here $p(C_m)$ and $q(C_m)$ are the distributions of C_m under $p(\mathbf{I}_m \mid \mathbf{B}_m)$ and $q(\mathbf{I}_m)$ respectively. Thus the model is in the form of foreground $p(C_m)$ popping out from background $q(\mathbf{I}_m)$. Specifically, $p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m)p(C_m)/q(C_m)$. Such a density substitution scheme was first used in projection pursuit density estimation [13], see also [15].

For computational simplicity, we further assume that $(c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle, i = 1, ..., n)$ are independent given $\mathbf{B}_m$, under both p and q, so

$$p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m) \prod_{i=1}^{n} \frac{p_i(c_{m,i})}{q(c_{m,i})},$$

where $q(c)$ is assumed to be the same for $i = 1, ..., n$ because $q(\mathbf{I}_m)$ is translation and rotation invariant. $q(c)$ can be pooled from natural images in the form of a heavy-tailed histogram of Gabor filter responses.

Exponential model. For parametric modeling, we assume the following exponential family model $p_i(c) = p(c; \lambda_i)$, where

$$p(c; \lambda) = \frac{1}{Z(\lambda)} \exp\{\lambda h(|c|^2)\} q(c). \tag{7}$$

$h(r)$ is a function of the response $r = |c|^2$ that saturates for large r. Specifically, we assume that $h(r) = \xi[2/(1 + e^{-2r/\xi}) - 1]$, so $h(r) \approx r$ for small r, and $h(r) \to \xi$ as $r \to \infty$ (default setting: $\xi = 6$). In (7),

$$Z(\lambda) = \int \exp\{\lambda h(r)\} q(c) dc = \mathrm{E}_q[\exp\{\lambda h(r)\}]$$

is the normalizing constant. $\mu(\lambda) = \mathrm{E}_\lambda[h(r)] = \int h(r) p(c; \lambda) dc$ is the mean parameter. Both $Z(\lambda)$ and $\mu(\lambda)$ can be computed beforehand from natural images.

The log-likelihood is

$$l(\{\mathbf{I}_m\} \mid \mathbf{B}, \{\mathbf{B}_m\}) = \sum_{m=1}^{M} \sum_{i=1}^{n} \left[\lambda_i h(\langle \mathbf{I}_m, B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}} \rangle) - \log Z(\lambda_i) \right]. \tag{8}$$

2.5 Shared matching pursuit

Learning. We revise the prototype algorithm in subsection (2.3) in order to maximize the log-likelihood (8) instead of minimizing the squared loss (5) as in subsection (2.3). The revised version of the shared matching pursuit algorithm is as follows.

[0] Initialize $i \leftarrow 0$. For $m = 1, ..., M$, initialize the response maps $R_m(x, \alpha) \leftarrow \langle \mathbf{I}_m, B_{x,s,\alpha} \rangle$ for all (x, α).

[1] $i \leftarrow i + 1$. Select the next basis function by finding

$$(x_i, \alpha_i) = \arg\max_{x,\alpha} \sum_{m=1}^{M} \max_{\Delta x, \Delta \alpha} h(|R_m(x + \Delta x, \alpha + \Delta \alpha)|^2),$$

where $\max_{\Delta x, \Delta \alpha}$ is again local maximum pooling.

[2] For $m = 1, ..., M$, given (x_i, α_i), infer the perturbations by retrieving the arg-max in the local maximum pooling of step [1]:

$$(\Delta x_{m,i}, \Delta \alpha_{m,i}) = \arg\max_{\Delta x, \Delta \alpha} |R_m(x_i + \Delta x, \alpha_i + \Delta \alpha)|^2.$$

Let $c_{m,i} \leftarrow R_m(x_i + \Delta x_{m,i}, \alpha_i + \Delta \alpha_{m,i})$, and update $R_m(x, \alpha) \leftarrow 0$ if $\mathrm{corr}[B_{x,s,\alpha}, B_{x_i + \Delta x_{m,i}, s, \alpha_i + \Delta \alpha_{m,i}}] > \epsilon$ (default setting: $\epsilon = .1$). Then compute λ_i by solving the maximum likelihood equation $\mu(\lambda_i) = \sum_{m=1}^{M} h(|c_{m,i}|^2)/M$.

[3] Stop if $i = n$, else go back to step [1].

The correlation is defined as the square of the inner product between the basis functions and can be stored beforehand. In step [2], the arg-max basis function inhibits nearby basis functions to enforce the approximate orthogonality constraint.

Inference. After learning the template from training images $\{\mathbf{I}_m\}$, we can use the learned template to detect the object in a testing image $\mathbf{I}$.

[1] For every pixel X, compute the log-likelihood $l(X)$, which serves as the template matching score at putative location X:

$$l(X) = \sum_{i=1}^{n} [\lambda_i \max_{\Delta x, \Delta \alpha} h(|\langle \mathbf{I}, B_{X+x_i+\Delta x, s, \alpha_i+\Delta\alpha}\rangle|^2) - \log Z(\lambda_i)]. \tag{9}$$

[2] Find maximum likelihood $\hat{X} = \arg\max_X l(X)$. For $i = 1, ..., n$, inferring perturbations by retrieving the arg-max in the local maximum pooling in step [1]:

$$(\Delta x_i, \Delta \alpha_i) = \arg \max_{\Delta x, \Delta \alpha} |\langle \mathbf{I}, B_{\hat{X}+x_i+\Delta x, s, \alpha_i+\Delta\alpha}\rangle|^2.$$

[3] Return the location $\hat{X}$, and the translated and deformed template $(B_{\hat{X}+x_i+\Delta x_i, s, \alpha_i+\Delta\alpha_i}, i = 1, ..., n)$.

Rotation and multi-resolution. We can rotate the template and scan the template over multiple resolutions of the original image, to account for uncertainties about the orientation and scale of the object in the testing image.

3 Compositional sparse code: model and algorithm

In the sparse coding model (1), the coefficients of the basis functions in the dictionary are assumed to be independent. A natural question is how to correct this assumption. Since we argue that the Olshausen-Field model with a dictionary of self-similar basis functions as in (2) is essentially a spatial point process as explicated in (3), we may search for compositional patterns in the spatial arrangements of the basis functions with non-zero coefficients.

In this section, we shall specify our compositional sparse coding model where each representational unit is an active basis model. Then we shall describe the algorithm for learning dictionaries of active basis models from training images.

3.1 Compositional sparse coding model

In this subsection, we strive to write down our model in a form that is analogous to the Olshausen-Field model (3), by using compactified notation.

Compactified notation. As the first step of this exercise of compactification, let us slightly generalize the active basis model by assuming that the template may appear at location X_m in image $\mathbf{I}_m$, then we can write the representation in the following form:

$$\begin{aligned} \mathbf{I}_m &= \sum_{i=1}^{n} c_{m,i} B_{X_m+x_i+\Delta x_{m,i}, s, \alpha_i+\Delta\alpha_{m,i}} + U_m \\ &= C_m \mathbf{B}_{X_m} + U_m, \end{aligned} \tag{10}$$

where $\mathbf{B}_{X_m} = (B_{X_m+x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$ is the deformed template spatially translated to X_m, $C_m = (c_{m,i}, i = 1, ..., n)$, and $C_m B_{X_m}$ is defined to be $\sum_{i=1}^n c_{m,i} B_{X_m+x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}$. Here we no longer assume that the training images $\{\mathbf{I}_m\}$ are aligned.

$\mathbf{B}_{X_m}$ explains the part of $\mathbf{I}_m$ that is covered by $\mathbf{B}_{X_m}$. For each image $\mathbf{I}_m$ and each X_m, we can define the log-likelihood ratio similar to (9):

$$\begin{aligned} l(\mathbf{I}_m \mid \mathbf{B}_{X_m}) &= \log \frac{p(\mathbf{I}_m \mid \mathbf{B}_{X_m})}{q(\mathbf{I}_m)} \\ &= \sum_{i=1}^n \left[\lambda_i \max_{\Delta x, \Delta\alpha} h(|\langle \mathbf{I}_m, B_{X_m+x_i+\Delta x,s,\alpha_i+\Delta\alpha}\rangle|^2) - \log Z(\lambda_i) \right]. \end{aligned} \tag{11}$$

As the next step of this compactification exercise, in addition to spatial translation and deformation, we can also rotate and scale the template. So a more general version of (10) is

$$\mathbf{I}_m = C_m \mathbf{B}_{X_m,S_m,A_m} + U_m,$$

where X_m is the location, S_m is the scale, and A_m is the orientation of the translated, rotated, scaled and deformed template. The scaling of the template is implemented by changing the resolution of the original image. We adopt the convention that whenever the notation $\mathbf{B}$ appears in image representation, it always means the deformed template, where the perturbations of the basis functions can be inferred by local max pooling. The log-likelihood ratio $l(\mathbf{I}_m \mid \mathbf{B}_{X_m,S_m,A_m})$ can be similarly defined as in (11).

Compactified representation. Now suppose we have a dictionary of T active basis templates, $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$, where each $\mathbf{B}^{(t)}$ is a compositional pattern of basis functions. Then we can represent the image $\mathbf{I}_m$ by K_m templates that are spatially translated, rotated, scaled and deformed versions of these T templates in the dictionary:

$$\mathbf{I}_m = \sum_{k=1}^{K_m} C_{m,k} \mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}} + U_m, \tag{12}$$

where each $\mathbf{B}^{(t_k)}_{X_{m,k},S_{m,k},A_{m,k}}$ is obtained by translating the template of type t_k, i.e., $\mathbf{B}^{(t_k)}$ in the dictionary, to location $X_{m,k}$, scale it to scale $S_{m,k}$, rotate it to orientation $A_{m,k}$, and deform it to match $\mathbf{I}_m$.

If the K_m templates do not overlap with each other, then the log-likelihood is

$$\sum_{m=1}^M \sum_{k=1}^{K_m} \left[l(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}) \right]. \tag{13}$$

Packing and unpacking. The above representation is in analogy to Equation (3) in subsection (2.1), which we copy here: $\mathbf{I}_m = \sum_{i=1}^n c_{m,i} B_{x_{m,i},s_{m,i},\alpha_{m,i}} + U_m$. The difference is that each $\mathbf{B}^{(t_k)}_{X_{m,k},S_{m,k},A_{m,k}}$ is a composite representational unit, which is itself a group of basis functions that

follow a certain compositional pattern of type t_k. Because of such grouping or packing, the number of templates K_m needed to encode $\mathbf{I}_m$ is expected to be much smaller than the total number of basis functions needed to represent $\mathbf{I}_m$, thus resulting in sparser representation. Specifically, if each template is a group of g basis functions, then the number of basis functions in the representation (12) is $K_m g$. In fact, we can unpack model (12) into the representation (3). The reason that it is advantageous to pack the basis functions into groups is that these groups exhibit T types of frequently occurring spatial grouping patterns, so that when we encode the image $\mathbf{I}_m$, for each selected group $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$, we only need to code the overall location, scale, orientation and type of the group, instead of the locations, scales and orientations of the individual constituent basis functions.

Connection with group Lasso. In the representation (12), each $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$ is a group of basis functions, and the K_m groups are to be selected from the collection of groups that correspond to all possible translated, rotated, scaled and deformed versions of the compositional patterns in the dictionary. The situation is very similar to that of group Lasso [33], which is also about selecting groups of variables from all possible groups. The selection by group Lasso is accomplished by solving a penalized least squares problem with a composite penalty based on the group structure. Our work goes beyond the group Lasso scenario in that the collection of groups is unknown, and we learn a dictionary of compositional patterns of these groups from training images. This dictionary then defines a large collection of groups by translation, rotation, scaling and deformation. Our work is a special case of structured sparsity. We call it compositional sparsity because we learn compositional patterns in sparse representations.

Limited overlap assumption. It is desirable to allow some overlaps between the bounding boxes of the K_m templates that encode $\mathbf{I}_m$, so that no salient structures of $\mathbf{I}_m$ fall through the cracks between the templates. Suppose the bounding boxes of all the templates are of the squared shape. We assume the following limited overlap constraint: For each template $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$ centered at $X_{m,k}$, let D be the side length of its squared bounding box, then no other templates are allowed to be centered within a distance of ρD from $X_{m,k}$ (default setting: $\rho = .4$). Under such limited overlap condition, we may continue to use the log-likelihood (13). However, we need to re-scale it to account for the overlap between the K_m templates.

3.2 Model complexity

Before considering the learning algorithm that seeks to maximize the log-likelihood (13), we need to resolve two issues regarding model complexity. One is how to choose the number of basis functions $n^{(t)}$ in each template $\mathbf{B}^{(t)}$ in the dictionary. The other is how to choose the number of templates T in the dictionary $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$.

Determining the number of basis functions in a template in supervised learning. Suppose we are in the supervised learning setting where $\{\mathbf{I}_m, m = 1, ..., M\}$ are aligned image patches, and we want to learn an active basis template $\mathbf{B} = \{B_{x_i,s,\alpha_i}, i = 1, ..., n\}$. We employ the following penalized log-likelihood:

$$\sum_{m=1}^{M} \sum_{i=1}^{n} \left[\lambda_i h(\langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle) - \log Z(\lambda_i) - \gamma\right], \tag{14}$$

which is the sum of the log-likelihood (8) and a penalty term $-\gamma$ associated with each basis function. There are two interpretations of γ. One is from the minimum description length (MDL) [27] perspective, where γ can be interpreted as the cost of coding the perturbations $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ in the encoding of $\mathbf{I}_m$. The other interpretation is from the Bayesian information criterion (BIC) [28] perspective. From the Bayesian perspective, the perturbations $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ should be integrated out according to their prior distributions, which are uniform distributions over the allowed ranges of perturbations. However, in our computation, the perturbations $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ are inferred by local max pooling, i.e., they are actually maxed out instead of being integrated out. As a result, the resulting log-likelihood with perturbations maxed out actually over-estimates the log-likelihood with perturbations integrated out. The term γ may be considered an approximation to this over-estimation, which should be subtracted from the maxed-out log-likelihood.

In order to maximize the penalized log-likelihood (14), we can continue to use the shared matching pursuit algorithm in subsection (2.5), except that we should stop the algorithm once the gain in the average log-likelihood is smaller than γ, i.e.,

$$\lambda_i \sum_{m=1}^{M} h(\langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}\rangle)/M - \log Z(\lambda_i) < \gamma.$$

This determines n. Currently we set the tuning parameter γ empirically at the default value of 1.

We can apply the above idea to unsupervised learning of the dictionary $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$ from the non-aligned training images $\{\mathbf{I}_m\}$, by maximizing the penalized log-likelihood function

$$\sum_{m=1}^{M} \sum_{k=1}^{K_m} \left[l(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}) - n^{(t_{m,k})}\gamma\right], \tag{15}$$

where $n^{(t)}$ is the number of basis functions in $\mathbf{B}^{(t)}$.

Determining the number of templates. The number of templates T in the dictionary can be selected by an adjusted BIC criterion. The BIC criterion has been advocated by Fraley and Raftery (2002) [12] for determining the number of clusters in mixture models. Our model is similar to mixture models or clustering in the sense that the image patches are clustered into T different clusters. The difference is that these image patches are not independent examples, nor are they

randomly cropped from the training images. Instead, they are to be cropped from the training images under the limited overlap constraint by a template matching pursuit algorithm to be described in the next subsection. From the clustering perspective, our method also bears some similarity to bi-clustering [22] or co-clustering, where in each cluster, different sets of basis functions are selected for representation. The difference is that these basis functions are not given variables. They need to be inferred from the image patches.

Recall that the BIC criterion is of the following form: maximized log-likelihood - number of parameters $\times$ log (number of training examples)/2. We can take (15) as the log-likelihood term, however, we have to discount the overlap between the templates. The number of parameters in our model is $\sum_{t=1}^{T} n^{(t)}$, which is the number of basis functions in the templates in the dictionary. The number of training examples can be taken as $\sum_{m=1}^{M} K_m$, which is the number of image patches that are explained by the templates. So we define our adjusted BIC criterion as:

$$\beta \sum_{m=1}^{M} \sum_{k=1}^{K_m} \Big[l(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}) - n^{(t_{m,k})}\gamma \Big] - \frac{1}{2} \sum_{t=1}^{T} n^{(t)} \log \sum_{m=1}^{M} K_m, \tag{16}$$

where β is a ratio that discounts the overlap between the selected templates $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}\}$. We define $\beta = a/b$ where a is the total number of pixels that are actually covered by the bounding boxes of these templates, where the overlapping pixels are only counted once. b is the sum of the numbers of pixels within the bounding boxes of these templates, where a pixel is counted multiple times if it is covered by multiple templates.

3.3 Unsupervised learning algorithm

Strategy 1: Sparsify and then compose. In order to discover the compositional patterns of basis functions in the sparse representations of the training images $\{\mathbf{I}_m\}$, it is tempting to first apply the plain matching pursuit or basis pursuit/Lasso to each training image $\mathbf{I}_m$ to obtain the sparse representation

$$\mathbf{I}_m = \sum_{i=1}^{n_m} c_{m,i} B_{x_{m,i},s_{m,i},\alpha_{m,i}} + U_m, \tag{17}$$

and then try to discover frequently occurring compositional patterns in the spatial grouping of the selected basis functions $\{B_{x_{m,i},s_{m,i},\alpha_{m,i}}, \forall i, m\}$. This was actually the strategy adopted by Zhu et al. [36] in their work on textons. The problem with such a sparsify-and-then-compose scheme is that the representation (17) produced by the matching pursuit or basis pursuit is an early decision or early commitment. Presumably, there may be many other representations that are no much less sparse than the representation (17), but they may be much more regular in terms of forming recurring compositional patterns. The results from matching pursuit or basis pursuit simply do

not account for such uncertainty. In fact, even if the selected basis functions do form recurring patterns, discovering these patterns requires reasoning the correspondences between different groups that correspond to the same pattern. This is not an easy task, especially if the numbers of basis functions in the groups are not very small. The learning algorithm described below avoids such a sparsify-and-then-compose strategy. Not relying on early decision on sparse coding or edge detection is a key difference between our learning algorithm and those of [11] and [35].

Strategy 2: Iterative encoding and re-learning. Our learning algorithm seeks to maximize the penalized log-likelihood (15) subject to the limited overlap constraint. It is an iterative algorithm where each iteration seeks to maximally increase the penalized log-likelihood (15). Each iteration consists of two steps in the EM-style [6]: (I) Image encoding. Given the current dictionary $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$, encode each training image $\mathbf{I}_m$ by translated, rotated, scaled and deformed versions of the templates in the dictionary, i.e., $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}, k = 1, ..., K_m\}$. (II) Dictionary learning. Given the current encoding of $\mathbf{I}_m$ by $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}, k = 1, ..., K_m\}$, re-learn each $\mathbf{B}^{(t)}$ from the image patches covered by the translated, rotated, scaled and deformed versions of the current $\mathbf{B}^{(t)}$, i.e., $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}, t_{m,k} = t\}$.

In the above learning process, we fix the total number of templates in the dictionary, T. We run the learning process for different values of T. Then we choose T that achieves the maximum of the adjusted BIC (16).

The following are the details of the two steps:

Step (I): Image encoding by template matching pursuit. Suppose we are given the current dictionary $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$. Then for each $\mathbf{I}_m$, the template matching pursuit process seeks to represent $\mathbf{I}_m$ by sequentially selecting a small number of templates from the dictionary. Each selection seeks to maximally increase the penalized log-likelihood (15).

[I.0] Initialize the maps of template matching scores for all (X, S, A, t):

$$\mathbf{R}^{(t)}_m(X, S, A) \leftarrow l(\mathbf{I}_m \mid \mathbf{B}^{(t)}_{X,S,A}) - n^{(t)}\gamma,$$

where $n^{(t)}$ is the number of basis functions in the t-th template in the dictionary and γ is a constant controling model complexity as explained above. This can be accomplished by first rotating the template $\mathbf{B}^{(t)}$ to orientation A, and then scanning the rotated template over the image zoomed to the resolution that corresponds to scale S. The larger the S is, the smaller the resolution is. Initialize $k \leftarrow 1$.

[I.1] Select the translated, rotated, scaled and deformed template by finding the global maximum of the response maps:

$$(X_{m,k}, S_{m,k}, A_{m,k}, t_{m,k}) = \arg \max_{X,S,A,t} \mathbf{R}^{(t)}_m(X, S, A).$$

[I.2] Let the selected arg-max template inhibit overlapping candidate templates to enforce limited overlap constraint. Let D be the side length of the bounding box of the selected template

$\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$, then for all (X, S, A, t), if X is within a distance ρD from $X_{m,k}$, then set the response $\mathbf{R}^{(t)}_m(X, S, A) \leftarrow -\infty$ (default setting: $\rho = .4$).

[I.3] Stop if all $\mathbf{R}^{(t)}_m(X, S, A, t) \leq 0$. Otherwise let $k \leftarrow k + 1$, and go to [I.1].

The template matching pursuit algorithm implements a hard inhibition to enforce the limited overlap constraint. In a more rigorous implementation, we may update the residual image by $U_m \leftarrow U_m - C_m \mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$ as in the original version of matching pursuit. But the current simplified version is more efficient and works well enough.

The regularization parameter γ play an important role in determining when to stop the template matching pursuit algorithm in the image encoding step (I). The response map $\mathbf{R}^{(t)}_m(X, S, A)$ is initialized as $l(\mathbf{I}_m \mid \mathbf{B}^{(t)}_{X,S,A}) - n^{(t)}\gamma$. If $l(\mathbf{I}_m \mid \mathbf{B}^{(t)}_{X,S,A}) < n^{(t)}\gamma$, then the gain in terms of the log-likelihood ratio does not compensate for the cost of coding the perturbations of the basis elements of the template. So we should stop the template matching pursuit if this is the case with all the remaining candidate templates.

Step (II): Dictionary re-learning by shared matching pursuit. For each $t = 1, ..., T$, we re-learns $\mathbf{B}^{(t)}$ from all the image patches that are currently covered by $\mathbf{B}^{(t)}$. Each iteration of the shared matching pursuit process seeks to maximally increase the penalized log-likelihood (15), given the current encoding $(t_{m,k}, X_{m,k}, S_{m,k}, A_{m,k}, k = 1, ..., K_m)$.

[II.0] Image patch cropping. For each $\mathbf{I}_m$, go through all the selected templates $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}, \forall k\}$ that encode $\mathbf{I}_m$. If $t_{m,k} = t$, then crop the image patch of $\mathbf{I}_m$ (at the resolution that corresponds to $S_{m,k}$) covered by the bounding box of the template $\mathbf{B}^{(t_{m,k})}_{X_{m,k},S_{m,k},A_{m,k}}$.

[II.1] Template re-learning. Re-learn template $\mathbf{B}^{(t)}$ from all the image patches covered by $\mathbf{B}^{(t)}$ that are cropped in [II.0], with their bounding boxes aligned. The learning is accomplished by the shared matching pursuit algorithm of subsection (2.5).

Figure 5 illustrates the learning of maple leaf template from the training image shown in Figure 2. Figure 5.(a) traces the template of maple leaf learned over the first 7 iterations of the learning algorithm. (b) shows the process of shared matching pursuit for learning this template in the last (10th) iteration, where the constituent basis functions are sequentially added.

Figure 5: (a) Template of leaf learned in the first 7 iterations of the unsupervised learning algorithm. (b) In each of iteration, the shared matching pursuit process selects wavelet elements sequentially to form each template. The sequence shows the process selecting 1, 3, 5, 10, 20, 30, 40 wavelets to form the leaf template in the last (10th) iteration.

This dictionary re-learning step re-learns each compositional pattern from the re-aligned raw

image patches, where the sparse representations and the correspondences between the selected basis functions are obtained simultaneously by the shared matching pursuit algorithm, thus avoiding the difficulty faced by the sparsify-and-then-compose procedure of Strategy I.

Random initialization and polarization. The learning algorithm is initialized by learning each $\mathbf{B}^{(t)}$ from image patches that are randomly cropped from $\{\mathbf{I}_m\}$, and these initial templates are rather meaningless. Meaningful templates emerge very quickly after a few iterations. See Figure 5.(a) for an illustration. In the beginning, the differences among the initial templates are small. However, as the algorithm proceeds, the small differences among the initial templates trigger a polarizing or specializing process, so that the templates become more and more different, and they specialize in encoding different types of image patches. One may start the algorithm multiple times and select the dictionary that achieves the maximum of the log-likelihood (15).

3.4 Notes on the learning algorithm

This subsection consists of some notes on various subtle aspects of the learning algorithm. More practically minded readers can jump to the next section for experimental results.

Generalized matching pursuit in both steps. Both the image encoding step (I) and the dictionary re-learning step (II) are generalizations of the matching pursuit algorithm. In fact, the whole learning algorithm can be viewed as an encoding algorithm, which seeks to automatically discover the recurring compositional patterns of the selected basis functions that are otherwise overlooked by the plain matching pursuit algorithm. The re-learning of each template can be viewed as encoding multiple image patches by a single template, thus resulting in more efficient encoding than the plain matching pursuit algorithm.

Relationship with sparse component analysis and K-SVD. As our model is a recursion of the Olshausen-Field model, our learning algorithm can also be viewed as a recursion of the learning scheme of Olshausen and Field, which is sometimes called sparse component analysis in the literature. In the Olshausen-Field model $\mathbf{I}_m = \sum_{i=1}^{N} c_{m,i} B_i + U_m$, the dictionary of $(B_i, i = 1, ..., N)$ is learned from the training image patches $\{\mathbf{I}_m\}$ by minimizing

$$\sum_{m=1}^{M} \left[\|\mathbf{I}_m - \sum_{i=1}^{N} c_{m,i} B_i\|^2 + \lambda \sum_{i=1}^{N} S(c_{m,i}) \right], \tag{18}$$

over both $(c_{m,i})$ and (B_i), where $S()$ is a sparsity inducing penalty function, and λ is the regularization parameter. The learning algorithm iterates the following two steps. (I) Image encoding. For each $\mathbf{I}_m$, update $(c_{m,i}, \forall i)$ given $(B_i, \forall i)$. (II) Dictionary learning. Update (B_i) given $(c_{m,i}, \forall i, m)$. In Olshausen-Field learning algorithm, both steps are carried out by gradient descent. In a related learning algorithm called K-SVD [1], (I) can be accomplished by any pursuit algorithm such as matching pursuit or basis pursuit, and (II) is accomplished by SVD. Our algorithm is even more

similar to K-SVD than to the Olshausen-Field algorithm. It is interesting to notice that in both K-SVD and our algorithm, the updating of each representational unit in step (II) is performed only on the image patches where this representational unit is currently active, i.e., the representational unit is re-learned from image patches that is currently encoded by this unit. Also similar to K-SVD, in our algorithm, the coefficients and the basis functions are updated together in dictionary re-learning in step (II).

Non-convex objective function. One complication about our learning method is that the negative log-likelihood is not convex, and our learning algorithm is a greedy algorithm that is similar to the EM algorithm [6]. In fact, this is also the case with the objective function (18) in Olshausen-Field sparse component analysis, which is non-convex in the joint domain of the unknown basis functions and their coefficients. It seems unlikely that a convex relaxations of the objective function can be found. This is also the case with many other unsupervised learning methods.

Linear subtraction versus occlusion. In both the template matching pursuit in step (I) and the shared matching pursuit in step (II), the explaining-away inhibition is carried out by hard inhibition that enforces limited overlap between templates in step (I) and the approximated non-overlap between basis functions in step (II). Such hard inhibition amounts to occlusion, where a selected template or basis function occludes nearby overlapping templates or basis functions. A more rigorous explaining-away mechanism in the context of linear additive structures (4) and (12) is by linear subtraction, where a selected template (group of basis functions) or basis function is linearly subtracted from the training images, see Equation (6), so that other templates or basis functions continue to explain the residual images. This linear subtraction scheme is more computationally demanding than hard inhibition. We shall investigate the more rigorous subtraction scheme in future work.

Matching pursuit versus penalized least squares. Both steps (I) and (II) in our learning algorithm are generalizations of matching pursuit, and both can be replaced by generalizations of basis pursuit or Lasso. Step (I) can be replaced by group Lasso, where the groups are the groups of basis functions that correspond to all possible translated, rotated, scaled and deformed versions of the templates in the current dictionary. Step (II) can be replaced by a different type of group Lasso, where the groups are formed across the aligned image patches from which we re-learn the template. Specifically, we group the coefficients of the same basis functions (up to local perturbations) across the aligned image patches, so that we always select the same set of basis functions for these aligned image patches. This is sometimes called support union recovery in multivariate regression [23]. Such penalized least squares schemes can be much more expensive computationally than the corresponding versions of matching pursuit. We shall investigate them in future work.

4 Experiments

This section presents experiments based on the unsupervised learning algorithm in the previous section. The data and code for reproducing the experimental results reported in this paper can be downloaded at http://www.stat.ucla.edu/∼ywu/ABC/ABC.html.

4.1 Image representation

In order to learn the dictionaries of compositional patterns from training images, we run the algorithm for 10 iterations. In the first iteration, we stop the template matching pursuit process for each $\mathbf{I}_m$ until all $\mathbf{R}_m^{(t)}(X, S, A)$ become $-\infty$. That is because the initial templates in the dictionary are rather random, so we force them to explain the whole image of $\mathbf{I}_m$ even if the templates do not match the image well. We fix $n^{(t)}$ to be the maximal value (default setting: 40) in the first 9 iterations. In the last iteration, we choose $n^{(t)}$ using the method described in subsection (3.2), and we use the templates with adaptively chosen $n^{(t)}$ to represent the training images.

Figure 6 shows an example of selecting the number of templates T in the dictionary. In each row, the first image is the training image. The remaining four blocks display the learned dictionaries of compositional patterns of basis functions in the form of active basis templates, as well as the representations of the training images using the learned dictionaries. The numbers of templates in the dictionaries are respectively 1, 2, 3, and 4. Just as in Figures 2 and 5, each basis function or wavelet is illustrated by a bar at the same location and orientation, and with the same length as the corresponding wavelet. All the templates are of the size 100×100. We also display the adjusted BIC criterion for each learned dictionary. The learned dictionaries are quite meaningful, and they give meaningful representations of the training images. It is interesting to observe how the dictionaries with only 1 template strive to represent the images.

As to the issue of selecting image resolution, for the example of maple leaves, we also learn dictionaries of templates at different resolutions of the training image. The resolution in Figure 6.(b) achieves the maximum BIC per pixel. This is essentially equivalent to determining the size of templates. However, we feel that instead of selecting the optimal resolution, it may be more appropriate to learn dictionaries at multiple resolutions or scales and combine them for image representation and understanding, just like the multi-resolution analysis in wavelets theory.

The adjusted BIC is useful for determining the number of compositional patterns in the dictionary. However, it may be more appropriate to use it to determine the rough range of possible numbers of patterns, instead of using it to pinpoint the exact number. For instance, in the maple leaves example, the dictionary with 3 patterns also give a very meaningful representation of the training image. For applications such as image classification, the number of patterns in the dictionary may be determined by cross validation instead of BIC.

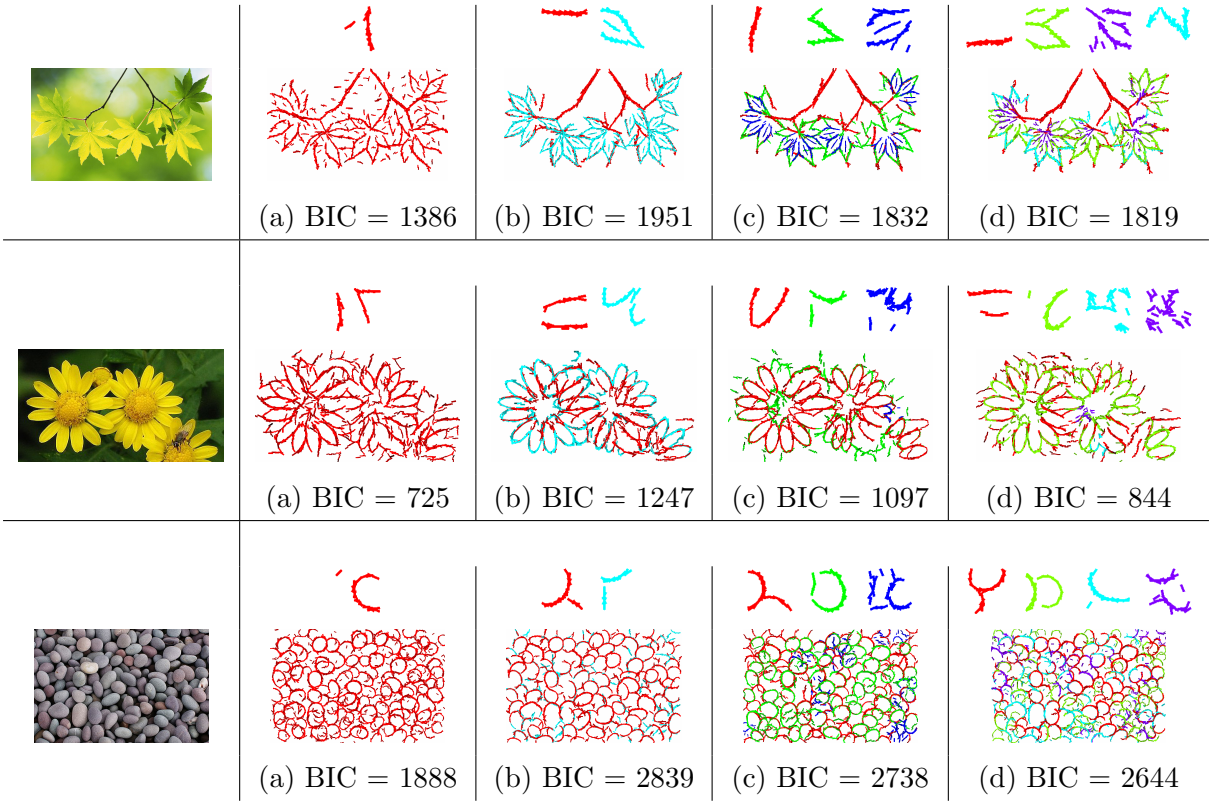


Figure 6: The adjusted BIC computed for different numbers of templates (1-4) in the dictionaries. The size of templates is 100 × 100. The allowed range of scale change is $\{.8, 1, 1.2\}$ of the original image. The templates are allowed full range of rotation. The maximal number of basis functions in each template is 40 and the actual number is automatically determined.

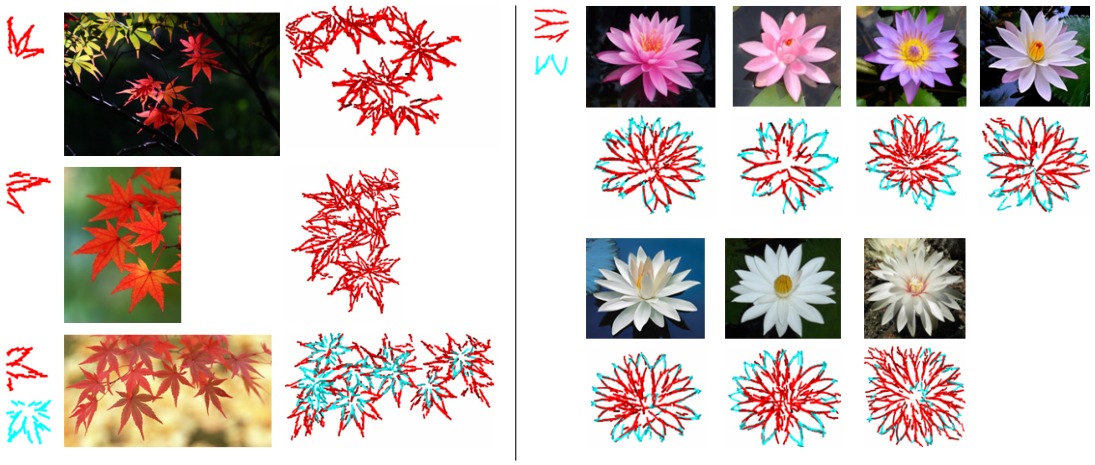


Figure 7: Maple leaves and lotus. Parameter setting is the same as in Figure 6.

Figure 8: Flowers. Parameter setting is the same as in Figure 6. For the two examples in the middle, the templates consist of more than one petal. This is due to the fact that the templates are of a squared shape and are relatively large.

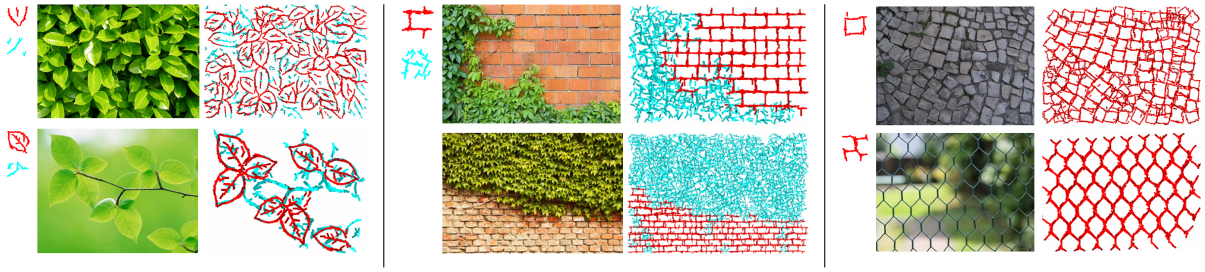


Figure 9: Ivy, leaves, ivy wall, pavement, and fence. Parameter setting is the same as in Figure 6. For the ivy wall example, the bottom row are testing image and its representation by the dictionary learned from the training image on the top row.

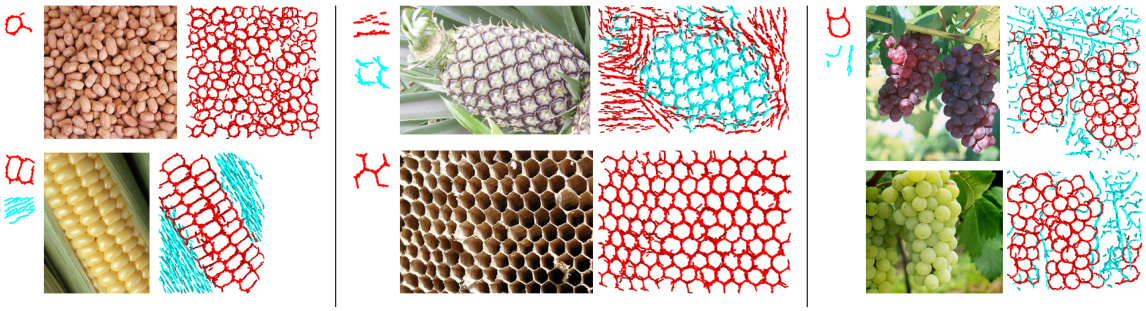


Figure 10: Peanuts, corn, pineapple, beehives, and grape. Parameter setting is the same as in Figure 6. For the grape example, the bottom row are testing image and its representation by the dictionary learned from the training image on the top row.

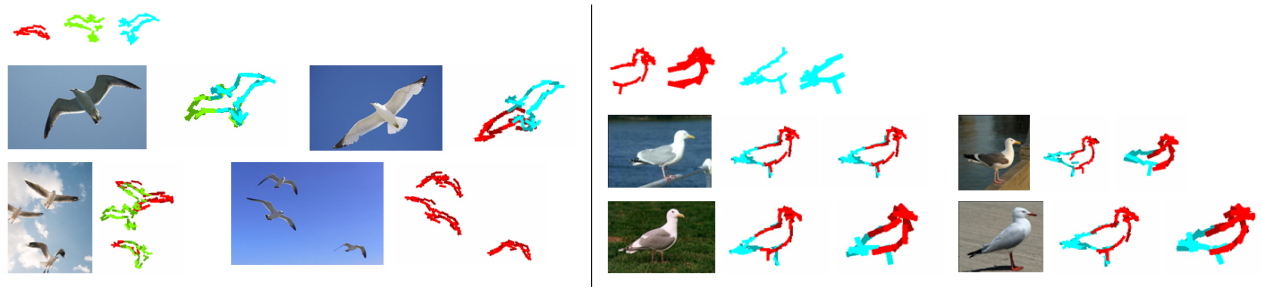


Figure 11: Seagulls flying (number of training images is 20) and standing (number of training images is 11). For the standing seagulls experiment, we learn multi-scale templates at two different scales (the scale parameters are .7 and 1.4 respectively). We sum the log-likelihood scores of the templates at the two scales in order to compute the template matching scores in the template matching pursuit process.

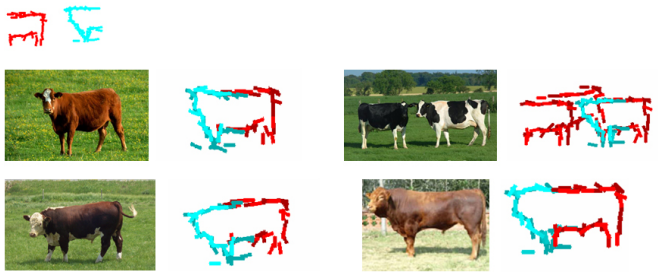


Figure 12: Cattle (number of training images is 17). Parameter setting is the same as in Figure 6 except that the allowed range of template rotation is $[-2, 2] \times \pi/16$.

Figure 13: Horse. Parameter setting is the same as in Figure 6, except that the allowed range of rotation of the template is $[-1, 1] \times \pi/16$.

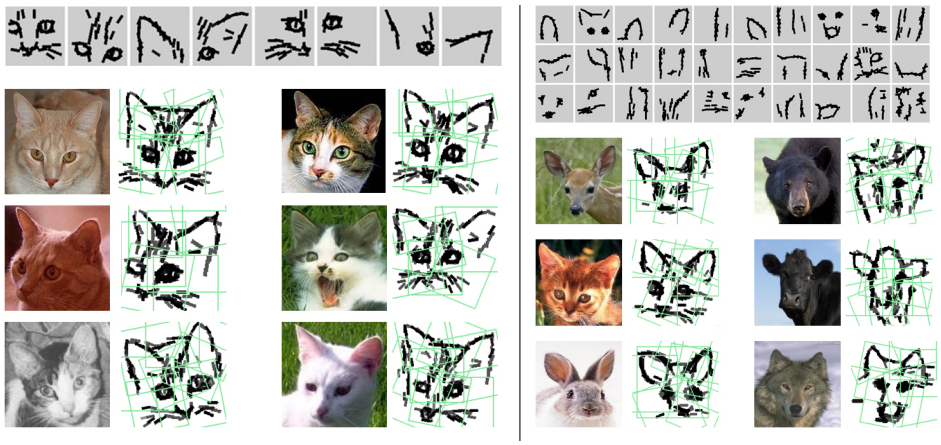


Figure 14: Cat faces (number of training images is 89) and animal faces (number of training images is 490). Parameter setting is the same as in Figure 12.

Figure 15: Cars. Parameter setting is the same as in Figure 12. Number of training images is 245.

Figures 7 to 15 show more examples of representing natural images. In some of the images, such as flowers, leaves and brick wall, the compositional patterns repeat themselves within the same image. For some other images, such as animal bodies and faces, the compositional patterns repeat across different images. For these images, the learning algorithm does not assume that the images are aligned.

In the dictionary re-learning step (II), for each entry in the dictionary, we can learn a multi-scale template from the aligned image patches using wavelets at multiple scales. A multi-scale template has multiple component templates, each consisting of wavelets at a single scale. The learning of each component template can be done separately at each scale. Then in the image encoding step (I), for each entry in the dictionary, we can combine the log-likelihood scores of the component templates at multiple scales in order to compute the overall template matching scores. See the standing seagulls experiment in Figure 11 for an illustration.

4.2 Better "words" for image classification

The learned compositional patterns can be used as "words" in the bag-of-word method for image classification. Let $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$ be the templates learned from positive training images. For each image $\mathbf{I}_m$, let $\mathbf{R}_m^{(t)}(X, S, A) = l(\mathbf{I}_m \mid \mathbf{B}^{(t)}_{X,S,A})$ be the log-likelihood score of $\mathbf{B}^{(t)}$ at location X, scale $S \in \{.8, 1, 1.2\}$ and orientation $A \in \{-1, 0, 1\} \times \pi/16$ (see subsection (3.1) and Equation (11)). Let $r_m^{(t)}(A) = \max(\max_{X,S} \mathbf{R}_m^{(t)}(X, S, A), 0)$ be the maximum score (lower-bounded by 0) at orientation A. Then for each $\mathbf{I}_m$, we have a $3T$-dimensional feature vector $(r_m^{(t)}(A), t = 1, ..., T, \forall A)$ (the factor 3 is due to the fact that we keep the scores of all the 3 orientations). We then train a linear logistic regression on such $3T$-dimensional feature vectors (regularized by ℓ_2 norm [8]) for image classification.

We evaluate this simple classifier on 16 categories from Caltech-101 [9], all ETHZ Shape [10] and all Graz-02 [21] data sets, where we test it on binary classification task. We resize all images to 150^2 pixels while maintaining their aspect ratios. We randomly sample 30 positive and negative images respectively as training data, and leave the rest as testing data. For Caltech-101 and Graz-02, negative images are from background category. For ETHZ, negative examples are from images other than the target category. For each category, we learn a dictionary of $T = 10$ templates. Each is of the size 100×100 with $n = 30$ basis functions.

As a comparison, for each image, we densely extract SIFT features [19] with patch size 16×16 and step size 8, from both positive and negative images, quantize them into 50, 100 and 500 words respectively by k-means clustering [4] and feed them into SVM [30, 2] (linear and histogram intersection kernel [16]). We take the best of these 6 results (3 numbers of words (50, 100, 500) $\times$ 2 types of SVM (linear, kernel)) and compare it with our method. All experiments are carried out with 5 independent runs and the 95% confident intervals on accuracies are calculated. Table

Table 1: Accuracies (%) on binary classification tasks for 24 categories from Caltech-101, ETHZ Shape and Graz-02 data sets.

Datasets	SIFT+SVM	Our method	Datasets	SIFT+SVM	Our method
Watch	90.1 ± 1.0	91.3 ± 2.0	Sunflower	76.0 ± 2.5	92.9 ± 2.5
Laptop	73.5 ± 5.3	87.9 ± 2.2	Chair	62.5 ± 5.0	89.1 ± 1.1
Piano	84.5 ± 4.2	93.4 ± 3.0	Lamp	61.5 ± 4.5	81.7 ± 3.7
Ketch	82.2 ± 0.8	89.2 ± 2.4	Dragonfly	66.0 ± 4.0	87.0 ± 4.1
Motorbike	93.9 ± 1.2	93.7 ± 0.9	Umbrella	73.4 ± 4.4	89.3 ± 2.5
Guitar	70.0 ± 2.4	80.9 ± 5.1	Cellphone	68.7 ± 5.1	87.9 ± 4.2
Schooner	64.3 ± 2.2	93.8 ± 2.7	Face	91.8 ± 2.3	95.8 ± 2.8
Ibis	67.8 ± 6.0	83.0 ± 1.9	Starfish	73.1 ± 6.7	85.3 ± 4.7
ETHZ-Bottle	68.6 ± 3.2	76.1 ± 3.3	ETHZ-Cup	66.0 ± 3.3	67.5 ± 4.4
ETHZ-Swans	64.2 ± 1.5	82.4 ± 0.5	ETHZ-Giraffes	61.5 ± 6.4	71.5 ± 3.5
ETHZ-Apple	55.0 ± 1.8	68.3 ± 5.2	Graz02-Person	70.4 ± 1.2	73.8 ± 2.3
Graz02-Car	64.0 ± 6.7	63.5 ± 5.1	Graz02-Bike	68.5 ± 2.8	77.6 ± 2.3

1 presents the results. It shows that our method generally outperforms the popular SIFT + SVM method even though our method uses a much smaller dictionary of words (10 in our method versus 50, 100, or 500 in SIFT + SVM).

We also test our method on the whole Caltech-101 data set. We learn $T = 200$ templates from the training images from all the 101 categories together. In one set of experiments, 15 training images are randomly sampled from each category in each run. In another set of experiments, 30 training images are randomly taken from each category in each run. In each set of experiments, 5 runs are repeated.

Each template is of the size 64×64 with $n = 15$ basis functions. During the learning algorithm, if the number of image patches encoded by a template is less than a threshold, then this template is eliminated from the dictionary. For 15 training images per category, the threshold is set at 5. For 30 training images per category, the threshold is set at 10. Figure 16 displays the learned dictionary of templates in one run of experiment with 30 training images per category. They seem to capture the mid-level structures such as lines, corners and circles etc.

For each image $\mathbf{I}_m$, and for each template $\mathbf{B}^{(t)}$ at each orientation A, besides the global maximum $r_m^{(t)}(A)$, we also divide $\mathbf{I}_m$ equally into 2×2 sub-regions and take the maximum within each sub-region. In addition to each maximum, we also take the corresponding average. So each $\mathbf{B}^{(t)}$ extracts 30 features from $\mathbf{I}_m$. Thus in total, each $\mathbf{I}_m$ produces a $30T$-dimensional feature vector. We adopt

Figure 16: Learned dictionary of templates from the Caltech-101 dataset in one run of an experiment, with 30 training images from each category. Template size is 64×64. Number of basis functions in each template is 15.

standard evaluation protocol. For 15 training images per category, the accuracy of our method is $61.6 \pm 2.2\%$ (compared to $57.7 \pm 1.5\%$ in [17]). For 30 training images per category, our result is $68.5 \pm 0.9\%$ (compared to $65.4 \pm 0.5\%$ in [17]). While more recent papers such as [34] and the references therein report better performances based on spatial pyramid matching [16], we do not use k-means to further cluster response maps into another layer of codewords, neither do we use any kernel.

5 Discussion

In this section, we first discuss the contributions and limitations of our current work. Then we shall compare our work with related work in the literature.

5.1 Contributions and limitations

The main contribution of this paper to propose a framework for learning compositional sparse code for representing natural images. We propose a representational scheme based on composite representational units, which are groups of basis functions of recurring compositional patterns, and we have developed an unsupervised learning algorithm for learning dictionaries of compositional patterns from training images, where the compositional patterns arise from seeking commonly shared sparse coding of image patches. Our experiments on natural images of plants and animals etc. show that our method is capable of learning meaningful compositional patterns, which lead to meaningful representations of training and testing images.

In terms of biological plausibility, the Olshausen-Field model is a model for simple V1 cells. The

Figure 17: Image scaling and regimes of patterns: As we zoom out the images, the patterns undergo a transition from low-entropy regime of geometric structures to mid-entropy regime of object shapes to high-entropy regime of stochastic textures. Our current method targets the mid-entropy regime of shape patterns.

local max pooling of Riesenhuber and Poggio [26] and the arg-max retrieval and inhibition of the active basis model may be related to complex V1 cells. The dictionaries of active basis templates learned by our method may be related to V2 cells and beyond.

The following are limitations of our work. First, as illustrated by Figure 17, as we zoom out the images, the image patterns undergo a transition from low-entropy regime of geometric structures to mid-entropy regime of object shapes to high-entropy regime of stochastic textures [31] (patterns such as the brick wall are also textures, but they are structural textures instead of stochastic textures). Our current model mainly targets the mid-entropy regime of object shapes and textons. It does not account for stochastic texture patterns or appearance patterns that are ubiquitous in natural scenes. The current model does not account for flatness patterns that are prominent in the low-entropy regime either. Another limitation is that the model is still not a fully generative model. We have not modeled the spatial arrangements of the templates, nor have we considered further composing the templates into yet another layer of composite representational units. A third limitation is that we assume that the dictionary of the basis functions are given as Gabor wavelets. It is desirable to learn these basis functions from training images by what may be called active component analysis.

The active basis models are currently learned by generative approach based on likelihood. It may be possible to learn the models discriminatively by regularized logistic regression after bringing in negative image patches, or to learn the models based on a combination of discriminative and generative loss functions.

5.2 Relations with hierarchical models

AND-OR grammar. Our work connects the sparsity principle to compositionality principle [14, 37, 15], which holds that the visual patterns are hierarchical compositions of constituent parts. In particular, in the language of AND-OR grammar of Zhu and Mumford (2006) [37], the dictionary

of the active basis templates can be considered a big OR node, where each template is a child node of this OR node, and each template is itself an AND-OR structure, where AND means composition of the constituent basis functions, and OR means perturbations of the locations and orientations of the basis functions, as well as variations of their coefficients. The OR-variations make the composite units invariant to their variations, so that the composite units are more abstract and generalizable.

In terms of composing Gabor wavelets, our representation is similar to that of [11] as well as [35]. The difference is that, our method is based on a top-down generative model, where the compositional patterns are re-learned in each iteration of the learning algorithm from raw image patches by seeking to maximize the likelihood. This enables us to compose many wavelets into a large template in a single layer, instead of recursively building up the templates via multiple layers.

Deep learning. Our work is related to deep learning with sparsity constraint [17, 34]. The difference is that our representational units are sparse compositions of automatically selected basis functions, where sparsity is explicitly built into the representational units by the shared matching pursuit process. The representational units are no longer linear basis functions on top of the wavelets coefficients or filter responses at the lower layer. Also, our model is not built on a pre-processed sparse representation, which, as we have argued in strategy I of subsection (3.3), amounts to early decision. In our learning algorithm, each iteration re-learns each template from raw image patches, where the sparse representations, their correspondences, and the template are obtained simultaneously instead of being obtained one after another.

HMAX model. Our work is related to the HMAX model of [26]. The local max pooling is employed for inferring the perturbations of the wavelet elements of the active basis model. In HMAX, the dictionary of the second layer consists of maps of local max pooling of Gabor responses. In our work, we explicitly learn the recurring compositional patterns of the wavelets guided by a generative model.

Acknowledgement

The work reported in this article is supported by NSF DMS 1007889, NSF IIS 1018715, ONR N00014-10-1-0933, and DARPA N00014-10-1-0933.

References

[1] M. Aharon, M. Elad, and A.M. Bruckstein. The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Transactions On Signal Processing*, **54**, 4311-4322, 2006.

[2] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 1-27, 2011.

[3] S. Chen, D. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, **20**, 33-61, 1999.

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *Workshop of European Conference on Computer Vision*, 2004.

[5] J. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of Optical Society of America*, **2**, 1160-1169, 1985.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, B*, **39**, 1-38, 1977.

[7] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, **47**, 2845-62, 2001.

[8] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, **9**, 1871-1874, 2008.

[9] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. *CVPR Workshop*, 2004.

[10] V. Ferrari, F. Jurie and C. Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, **87**, 284–303, 2010.

[11] S. Fidler, M. Boben, and A. Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[12] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, **97**, 611-631, 2002.

[13] J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, **82**, 249-266, 1987.

[14] S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, **60**, 707–736, 2002.

[15] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *International Conference on Machine Learning*, 2009.

[18] C. Lo, *Amazing Chinese Characters*, Panda Media Co., 2002.

[19] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**, 91–110, 2004.

[20] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, **41**, 3397-3415, 1993.

[21] M. Marszalek and C. Schmid. Accurate object localization with shape masks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[22] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

[23] G. Obozinski, M. J. Wainwright, and M. I. Jordan, Support union recovery in high-dimensional multivariate regression, *Annals of Statistics*, **39**, 1-47, 2011.

[24] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, **381**, 607-609, 1996.

[25] B. A. Olshausen, P. Sallee and M. S. Lewicki. Learning sparse image codes using a wavelet pyramid architecture. *Advances in Neural Information Processing Systems*, **13**, 887-893, 2001.

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

[27] J. Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.

[28] G. E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, **6**, 461-464, 1978.

[29] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B*, **58**, 267-288, 1996.

[30] V. N. Vapnik. *The nature of Statistical Learning Theory*. Springer, 2000.

[31] Y. N. Wu, C. Guo, S. C. Zhu, From information scaling of natural images to regimes of statistical models. *Quarterly of Applied Mathematics*, **66**, 81-122, 2008.

[32] Y. N. Wu, Z. Si, H. Gong, and S. C. Zhu. Learning active basis model for object detection and recognition. *International Journal of Computer Vision*, **90**, 198-235, 2010.

[33] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, B*, **68**, 49-67, 2006.

[34] M. Zeiler, G. Taylor and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. *International Conference on Computer Vision*, 2011.

[35] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised structure learning: hierarchical recursive composition, suspicious coincidence and competitive exclusion. *European Conference on Computer Vision*, 2008.

[36] S. C. Zhu, C. Guo, Y. Wang, and Z. Xu. What are textons? *International Conference on Computer Vision*, **62**, 121–143, 2005

[37] S. C. Zhu and D. B. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, **2**, 259–362, 2006.