Learning Codebooks of Deformable Templates for Sparse Image Representation

Anonymous CVPR submission

Paper ID 556

Abstract

This paper proposes a method for unsupervised learning of codebooks of deformable templates for sparse representations of images of various objects and textures. Each deformable template is represented by an active basis model, which is a composition of a small number of Gabor wavelets automatically selected from a dictionary of such wavelets. The selected wavelets are allowed to perturb their locations and orientations so that the active basis template can deform to encode the observed images. For a given set of training images, our method is to learn a codebook of such active basis templates, so that each training image can be represented by a small number of templates automatically selected from the codebook. The learning algorithm iterates the following two steps. (1) Image encoding by a template matching pursuit algorithm. (2) Codebook re-learning by a shared matching pursuit algorithm. Our experiments show that the method is capable of learning meaningful codebooks from images of textures and objects, and the codebooks give meaningful representations of these images. Our experiments also show that the learned codebooks can be used as features for classification.

1. Introduction

Learning codebooks or dictionaries of representational elements for images of natural scenes or images of various types of textures and objects is one of the most fundamental problems in both computer and biological vision. This paper proposes a method for unsupervised learning of codebooks of deformable templates that give sparse representations of images.

1.1. Explicit codebook and representation

Our method is based on an explicit generative model of image intensities, where each image is represented by a small number of templates automatically selected from a codebook of templates. Each template in the codebook is further encoded by a small number of Gabor wavelet elements automatically selected from a dictionary of Gabor wavelets. The codebook is learned from training images without any labelling or annotation. We also require that the number of templates in the codebook to be as small as possible.

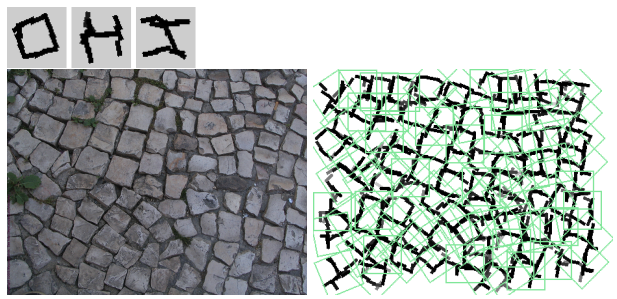

Figure 1. Learning a codebook of 3 active basis templates from an image of pavement. *Each black bar represents a Gabor wavelet at the same location, orientation and length*. The size of the image is 578 by 434 pixels. Each active basis template (i.e. entry in the codebook) is of size 60 by 60 pixels, with no more than 10 selected wavelets. The input image is paved by 43 deformed templates selected from the codebook, with their bounding boxes shown in green color.

Figure 1 illustrates the basic idea. The first row displays a codebook of 3 templates learned from a pavement image on the left of the second row. Each template in the codebook is represented by an active basis model. Specifically, each template is a composition of a small number of Gabor wavelets automatically selected from a dictionary of Gabor wavelets. *Each selected Gabor wavelet is depicted by a bar that has the same location, orientation and length as the Gabor wavelet*. The selected Gabor wavelets are allowed to perturb their locations and orientations, so the template becomes deformable, and the linear basis formed by the selected Gabor wavelet elements becomes active. The sketch on the right of the second row displays the representation of the original pavement image using the templates in the codebook. Here the word "pavement" also serves as a good metaphor because the representation is essentially a "pavement" made of the active basis templates. Specifically, each of the templates in the sketch is a *spatially trans-*

lated, scaled, rotated and deformed version of a template in the codebook. The bounding box of each template in the sketch is also shown in green color (the deformed templates are allowed to have some overlaps with each other). As shown in the caption, the image can be paved by a relatively small number of templates, and such a representation is said to be sparse. It is sparser than wavelet representation because the coding elements are compositions or groups of wavelets.

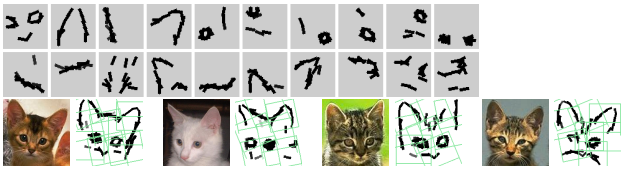

Figure 2. Learning a codebook of 20 active basis templates from 25 images of cats (not assumed to be aligned). The input images are resized to have roughly 22,500 pixels. Each active basis template (i.e. entry in the codebook) is of size 72 by 72 pixels, with no more than 12 selected wavelets. On average about 8 templates are used to sketch each image.

Figure 2 shows another example, where a codebook of active basis templates is learned from a set of training images of cats.

1.2. Unsupervised learning algorithm

For a given set of training images without any labeling or annotation, our learning algorithm iterates the following two steps:

Step 1: Image encoding by template matching pursuit: Given the current codebook of templates, for each training image, a template matching pursuit algorithm sequentially selects the templates from the codebook to pave the whole image.

Step 2: Codebook re-learning by shared matching pursuit: Given the current encoding, for each template, a shared matching pursuit algorithm re-learns the template from all the image patches that are currently covered by this template. The algorithm sequentially selects the constituent Gabor wavelet elements of the template from the dictionary of Gabor wavelets, in order to encode all these patches simultaneously.

Our experiments show that our method is capable of learning meaningful codebooks from images of textures and objects, and the codebooks give meaningful representations of these images. Our experiments also show that the learned codebooks can be used as features for classification.

1.3. Past work

Our generative model is built on the wavelet sparse coding model of Olshausen and Field [6] and the active basis model of Wu et al. [10]. In Olshausen-Field model, an image is represented by a small number of wavelet elements selected from a dictionary. In our model, an image is represented by a smaller number of groups of wavelet elements, and these groups of wavelets exhibit recurring compositional patterns that can be represented by active basis models.

The part of our work on textons can be considered a continuation of the work Zhu et al. [12]. The learned codebooks can serve as Or-nodes in the And-Or graph studied by Zhu and Mumford [13]. The learning method is also related to L. Zhu et al. [11].

2. Olshausen-Field model

Olshausen and Field [6] propose that the role of simple V1 cells is to provide sparse representations of natural images. This section reviews the Olshausen-Field model in order to set the stage and fix the notation.

2.1. Overcompleteness and sparsity

Let $\{\mathbf{I}_m, m = 1, ..., M\}$ be a set of training image patches (e.g. 12×12), 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 or codebook of basis elements of the same dimensionality 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 dimensionality of $\mathbf{I}_m$ (e.g. $N = 2 \times 12 \times 12$), so the dictionary is said to be overcomplete. 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 very small for each image $\mathbf{I}_m$. The dictionary of $(B_i, i = 1, ..., N)$ can be learned automatically from the training images $\{\mathbf{I}_m\}$ by imposing a sparsity regularization function.

2.2. Self-similarity and geometric attributes

One may also assume that the dictionary of the basis elements are translated, rotated and dilated version of one another, as in Olshausen et al. [7], 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 element, e.g. $B = B_{x=0,s=1,\alpha=0}$, and (x, s, α)

are properly discretized. B can be learned from training images $\{\mathbf{I}_m\}$.

Assumption on wavelets in this paper. From now on, we assume that the dictionary of wavelets is self-similar, and $(B_{x,s,\alpha}, \forall(x,s,\alpha))$ is already available. 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 $cB = c_0B_0 + c_1B_1$. For projection $\langle \mathbf{I}, B\rangle = (\langle \mathbf{I}, B_0\rangle, \langle \mathbf{I}, B_1\rangle)$, and $|\langle \mathbf{I}, B\rangle|^2 = \langle \mathbf{I}, B_0\rangle^2 + \langle \mathbf{I}, B_1\rangle^2$.

2.3. 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 wavelet elements 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 elements, but here i only runs through the n selected basis elements instead of all the N basis elements as in (1)).

3. Active basis model

The active basis model was proposed by Wu et al. [10] for modeling deformable compositional patterns of the selected wavelet elements.

3.1. A single template for aligned image patches

Suppose we have a set of training image patches $\{\mathbf{I}_m, m = 1, ..., M\}$. This time they are defined on the same bounding box. The objects in these images come from the same category. They appear at the same location, scale and orientation, and in the same pose within the bounding box. 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 original 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 template for encoding $\mathbf{I}_m$, where $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ are the perturbations in the location and orientation respectively, in order to account for deformation. Both $\Delta x_{m,i}$ and $\Delta\alpha_{m,i}$ are assumed to vary within limited ranges (e.g. $\Delta x_{m,i} \in [-3, 3]$ pixels, and $\Delta\alpha_{m,i} \in [-\pi/16, \pi/16]$).

3.2. Shared matching pursuit: local maximum pooling and arg-max explaining-away inhibition

The selection of B_{x_i,s,α_i} and the inference of its perturbed versions $B_{x_i+\Delta x_{m,i}, s, \alpha_i + \Delta\alpha_{m,i}}$ can be accomplished by a shared matching pursuit algorithm that greedily minimizes $\sum_{m=1}^{M} |\mathbf{I}_m - \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i}, s, \alpha_i+\Delta\alpha_{m,i}}|^2$ (recall that the wavelet elements 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 element 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 $U_m \leftarrow U_m - c_{m,i} B_{x_i+\Delta x_{m,i}, s, \alpha_i+\Delta\alpha_{m,i}}$.

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

The above algorithm is a generalization of the matching pursuit algorithm of Mallat and Zhang [5]. In step [1], the $\max_{\Delta x, \Delta\alpha}$ is the local maximum pooling, proposed by Riesenhuber and Poggio [8] as a function of V1 complex cells. The selected B_{x_i,s,α_i} is supposed to encode all the images $\{\mathbf{I}_m\}$ simultaneously, subject to local perturbations. That is, B_{x_i,s,α_i} is shared by all the images. After the selection of the shared wavelet element B_{x_i,s,α_i}, we infer its perturbations by retrieving the arg-max of the local maximum pooling. This arg-max wavelet element $B_{x_i+\Delta x_{m,i}, s, \alpha_i+\Delta\alpha_{m,i}}$ then explains away a small part from U_m, thereby implicitly inhibits nearby wavelet elements from being selected in the future.

Assumption on orthogonality in this paper. Because of the arg-max explaining-away inhibition, the wavelet elements in 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 overlaps with each other. So from now on, we shall assume that these wavelet elements are orthogonal to each other, so that the coefficient $c_{m,i} = \langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i}, s, \alpha_i+\Delta\alpha_{m,i}}\rangle$. We write $C_m = (c_{m,i}, i = 1, ..., n)$. In practice, we allow small overlaps between the elements of $\mathbf{B}_m$.

3.3. Statistical modeling: foreground pops out from natural image background

The above algorithm implicitly assumes that the residual U_m is Gausian white noise. This assumption is wrong. A

better assumption is to assume that U_m follows the same distribution as that of natural images. Specifically, 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 $C_m = (c_{m,i}, i = 1, ..., n)$ 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 $q(\mathbf{I}_m)$ respective. We should find $\mathbf{B}$ and perturb it to $\{\mathbf{B}_m\}$ so that $p(C_m)$ and $q(C_m)$ have the maximum contrast in terms of the Kullback-Leibler divergence. 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 [3].

For computational simplicity, we further assume that $(c_{m,i}, i = 1, ..., n)$ are independent given $\mathbf{B}_m$, under both p and q, so we have

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

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

For parametric modeling, we assume the following exponential family model,

$$p_i(c) = \frac{1}{Z(\lambda_i)} \exp\{\lambda_i h(|c|^2)\} q(c), \tag{5}$$

where $h(r)$ is a function of the response $r = |c|^2$ that saturates for large r. Specifically, $h(r) = \xi[2/(1 + e^{-2r/\xi}) - 1]$. $h(r)$ behaves like $h(r) \approx r$ for small r, but $h(r) \to \xi$ (e.g., $\xi = 6$) as $r \to \infty$. The reason we assume this saturation function is that in the natural image background, there are also occasional (albeit less frequent) edges that give equally large responses r as those in foreground, so the probability ratio $p_i(c)/q(c)$ should go to a positive constant instead of 0 for large $|c|^2$. $Z(\lambda) = \int \exp\{\lambda h(r)\} q(c) dc = \mathrm{E}_q[\exp\{\lambda h(r)\}]$ is the normalizing constant. $\mu(\lambda) = \mathrm{E}_\lambda[h(r)]$ is the mean parameter.

3.4. Shared matching pursuit revised: saturation and hard inhibition

We can revise the shared matching pursuit in section (3.2) in order to maximize the log-likelihood instead of minimize the squared loss as in section (3.2). The 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 wavelet element 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 $\mathrm{corr}[B_{x,s,\alpha}, B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}] > \epsilon$. Then estimate $\hat{\lambda}_i = \mu^{-1}(\sum_{m=1}^{M} h(|c_{m,i}|^2)/M)$.

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

There are two modifications to the original shared matching pursuit in section (3.2). (1) In step [1], we apply the saturation function $h()$ to the response. This is justified by maximum likelihood based on the exponential family model (5). Intuitively, it means we discount very large responses, because there can also be occasional (though less frequent) large responses caused by the edges in natural image background. (2) In step [2], the arg-max element $B_{x_i+\Delta x_{m,i},s,\alpha_i+\Delta\alpha_{m,i}}$ directly inhibits nearby elements whose correlation with it is greater than a tolerance ϵ, instead of explaining away from U_m and inhibiting nearby elements indirectly. The correlation is defined as the square of the inner product. This hard inhibition is to approximately enforce the orthogonality assumption in section (3.2). *n may be adaptively chosen by setting a threshold on $\hat{\lambda}_i$.*

3.5. Template matching and shape filter

After learning the template from training images $\{\mathbf{I}_m\}$, for a testing image $\mathbf{I}$ that contains an instance of the object, we can detect it by scanning the template over the whole image and compute the template matching score.

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

$$\sum_{i=1}^{n} \left[\lambda_i \max_{\Delta x,\Delta\alpha} h(|\langle \mathbf{I}, B_{X+x_i+\Delta x,s,\alpha_i+\Delta\alpha} \rangle|^2) - \log Z(\lambda_i)\right]. \tag{6}$$

[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 $(B_{\hat{X}+x_i+\Delta x_i,s,\alpha_i+\Delta\alpha_i}, i = 1, ..., n)$.

$l(\hat{X})$ can also be used for classification.

Shape filter. The template $\mathbf{B}$ can be viewed as a shape filter, and the template matching score $l(X)$ is the filter response at location X. We may even write $\mathbf{B} * \mathbf{I}(X) = l(X)$. Step [2] finds the object and deforms the template. Step [3] not only returns a bounding box, but also sketches the detected object by the deformed template.

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

4. Unsupervised learning of codebooks of active basis templates

In Olshausen-Field model (1), the coefficients are assumed to be independent for simplicity. A natural question is how to correct this assumption, or specifically, how to add another layer of model on all the coefficients?

Since we argue that the Olshausen-Field model (1) with a dictionary of self-similar wavelets as in (2) is essentially a spatial point process as explicated in (3), the model on top of all the coefficients should focus on the geometric patterns formed by the wavelet elements with non-zero coefficients. In particular, we may search for recurring compositional patterns of the selected wavelet elements. These compositional patterns can be modeled by active basis models.

In this section, we shall specify our representation where each representational element is an active basis template. Then we shall describe the algorithm for learning codebooks of active basis templates from training images.

4.1. Representation

In this section, we strive to write down our model in a form that is analogous to the Olshausen-Field model, 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

$$\begin{aligned} \mathbf{I}_m &= \sum_{i=1}^{n} c_{m,i} B_{X_m + x_i + \Delta x_{m,i}, s, \alpha_i + \Delta\alpha_{m,i}} + U_m \\ &= C_m \mathbf{B}_{X_m} + U_m, \end{aligned} \tag{7}$$

where $\mathbf{B} = (B_{x_i,s,\alpha_i}, i = 1, ..., n)$ is the original template, $\mathbf{B}_{X_m} = (B_{X_m + x_i + \Delta x_{m,i}, s, \alpha_i + \Delta\alpha_{m,i}}, i = 1, ..., n)$ is the deformed template spatially translated to X_m. Equation (7) is written in what we call the compactified notation.

For each image $\mathbf{I}$ and each X, we can also define

$$\langle \mathbf{I}, \mathbf{B}_X \rangle = \sum_{i=1}^{n} \big[\lambda_i \max_{\Delta x, \Delta\alpha} h(|\langle \mathbf{I}, B_{X + x_i + \Delta x, s, \alpha_i + \Delta\alpha} \rangle|^2) - \log Z(\lambda_i) \big], \tag{8}$$

which is $l(X) = \mathbf{B} * \mathbf{I}(X)$ in section (3.5).

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

We can also define $\langle \mathbf{I}, \mathbf{B}_{X,S,A} \rangle$ in a similar way as in equation (8).

Compactified representation. Now suppose we have a codebook of T templates, $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$. Then we can represent an image by K templates that are spatially translated, rotated, scaled and deformed versions of these T templates in the codebook.

$$\mathbf{I}_m = \sum_{k=1}^{K} C_{m,k} \mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}} + U_m, \tag{9}$$

where each $\mathbf{B}^{(t_k)}_{X_{m,k}, S_{m,k}, A_{m,k}}$ is obtained by translating the template $\mathbf{B}^{(t_k)}$ in the codebook to location $X_{m,k}$, scale it to scale $S_{m,k}$, rotate it to orientation $A_{m,k}$, and deform it to match $\mathbf{I}_m$. Again, the scaling of a template can be implemented by changing the resolution of the image.

Packing and unpacking. The above representation is in analogy to equation (3) in section (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 itself a group of wavelet elements that follow a certain composition pattern t_k. Because of such grouping or packing, the number of templates K needed to code $\mathbf{I}_m$ is expected to be smaller than n. Specifically, if each template is a group of g wavelet elements, then $n = Kg$ (if there is no overlaps between templates). In other words, we can unpack model (9) into the wavelet expansion model (3). The reason that it is advantageous to pack the wavelets into groups is that these groups exhibit T types of recurring compositional patterns to be discovered automatically by the learning algorithm.

4.2. Learning algorithm

We want to learn the codebook $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$ from training images $\{\mathbf{I}_m, m = 1, ..., M\}$, and at the same time, infer the representation for each $\mathbf{I}_m$ in terms of the model (9). The objective function is the log-likelihood

$$\sum_{m=1}^{M} \sum_{k=1}^{K} \langle \mathbf{I}_m, \mathbf{B}^{(t_{m,k})}_{X_{m,k}, S_{m,k}, A_{m,k}} \rangle, \tag{10}$$

which is the sum of the log-likelihood of all the selected templates (recall that we should not use the simple squared

loss, which implicitly and incorrectly assumes that U_m is white noise, see section (3.4)). K is assumed to be small and may vary for different $\mathbf{I}_m$.

The learning algorithm is guided by the above objective function. It iterates an image encoding step and a codebook revision step.

Step 1: Image encoding by template matching pursuit. Suppose we are given the current codebook $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$. Then for each $\mathbf{I}_m$, $m = 1, ..., M$, the template matching pursuit algorithm seeks to represent it by sequentially selecting a small number of templates from the codebook in order to pave $\mathbf{I}_m$.

[0] Initialize the response maps $\mathbf{R}_m^{(t)}(X, S, A) \leftarrow \langle \mathbf{I}_m, \mathbf{B}_{X,S,A}^{(t)} \rangle$, for all (X, S, A, t). This can be accomplished by first rotating the template to orientation A, and then scanning the rotated template over the image zoomed to the resolution corresponds to scale S. The larger the S is, the smaller the resolution is. Let $k \leftarrow 1$.

[1] Select the translated, rotated, scaled and deformed template by finding the global maximum of the response maps: $(X_{m,k}, S_{m,k}, A_{m,k}, t_{m,k}) = \arg\max_{X,S,A,t} \mathbf{R}_m^{(t)}(X, S, A)$.

[2] Let the selected arg-max template inhibits overlapping candidate templates, i.e., for all (X, S, A, t), if $\text{corr}[\mathbf{B}_{X,S,A}^{(t)}, \mathbf{B}_{X_{m,k},S_{m,k},A_{m,k}}^{(t_{m,k})}] > \epsilon$, then set the response $\mathbf{R}_m^{(t)}(X, S, A) \leftarrow -\infty$. $k \leftarrow k + 1$.

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

This template matching pursuit algorithm implements a hard inhibition. The $\text{corr}[\mathbf{B}_1, \mathbf{B}_2]$ here is simply defined as the overlap between the bounding boxes of the two templates $\mathbf{B}_1$ and $\mathbf{B}_2$. It can also be defined in a more refined manner in terms of the inner products between the constituent elements of $\mathbf{B}_1$ and $\mathbf{B}_2$. More rigorously, we could even update the residual image by $U_m \leftarrow U_m - C_m \mathbf{B}_{X_{m,k},S_{m,k},A_{m,k}}^{(t_{m,k})}$ as in the original version of matching pursuit. But the current simplified version is more efficient and works well enough.

Step 2: Codebook re-learning by shared matching pursuit. For each $t = 1, ..., T$ in the codebook, the shared matching pursuit algorithm re-learns $B^{(t)}$ from all the image patches that are covered by $\mathbf{B}^{(t)}$ by selecting the constituent Gabor wavelet elements of $\mathbf{B}^{(t)}$. This consists of the following two operations:

[1] Image patch cropping. For each $\mathbf{I}_m$, go through all the selected templates $\{\mathbf{B}_{X_{m,k},S_{m,k},A_{m,k}}^{(t_{m,k})}, \forall k\}$ that pave $\mathbf{I}_m$. 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 $\mathbf{B}_{X_{m,k},S_{m,k},A_{m,k}}^{(t_{m,k})}$.

[2] Template re-learning. Re-learn template $\mathbf{B}^{(t)}$ from all the image patches covered by $\mathbf{B}^{(t)}$ that are cropped in step [1], and with their bounding boxes aligned. The learning is accomplished by the shared matching pursuit algorithm of section (3.4), which sequentially selects Gabor wavelet elements to encode all these image patches simultaneously.

In practice, we do not have to deal with these image patches directly. We only need to crop the maps of Gabor filter responses, and feed them into the shared sketch algorithm.

Polarization or specialization. The learning algorithm starts from a codebook of templates learned from randomly cropped image patches. As a result, the initial templates are rather random and meaningless, and the differences among them are small. However, as the algorithm proceeds, the small differences among the initial templates quickly start a polarizing or specializing process, where the templates become more and more different, and they specialize in coding different types of image patches. One may start the algorithm multiple times and select the one that achieves the maximum of the log-likelihood (10).

Generalized matching pursuit in both steps. Both the encoding and re-learning steps of the learning algorithm are generalizations of matching pursuit. In fact, the whole learning algorithm can be viewed as an encoding algorithm, which seeks to automatically discover the recurring compositional patterns 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.

No early decision on wavelet representation For learning the codebooks, it is tempting to first apply the plain matching pursuit algorithm to each training image, and then search for recurring compositional patterns in the selected wavelets produced by the plain matching pursuit. The problem with such a two-step sequential scheme is that the wavelet representation produced by the plain matching pursuit is an early decision or early commitment. Presumably, there may be many other wavelet representations that are equally good or comparable in terms of sparsity, but they may be much more regular in terms of forming recurring compositional patterns. So we have to obtain the wavelet representation and discover the compositional patterns simultaneously or by an iterative algorithm. Not making early decision on wavelet expansion or edge detection is a key difference between our learning algorithm and those of Zhu et al. [12] and L. Zhu et al. [11].

Biological plausibility. The Olshausen-Field model is a model for simple V1 cells. The local max pooling of Riesenhuber and Poggio [8] and the arg-max retrieval and inhibition of the active basis model may be related to complex V1 cells. The codebooks of active basis templates may be related to V2 cells.

5. Experiments

This section presents experimental results that show that our learning algorithm is capable of learning explicit and meaningful codebooks that give explicit and meaningful representations of training images.

5.1. Learning codebooks of textons

We first apply the learning algorithm to texture images, where we learn codebooks of active basis templates that can be used to pave the texture images. The learned templates are recurring patterns within the texture images, and we may call them textons, following Julesz. Figure 3 shows some examples. *Some members in the codebook may disappear if they fail to be selected in the encoding step.*

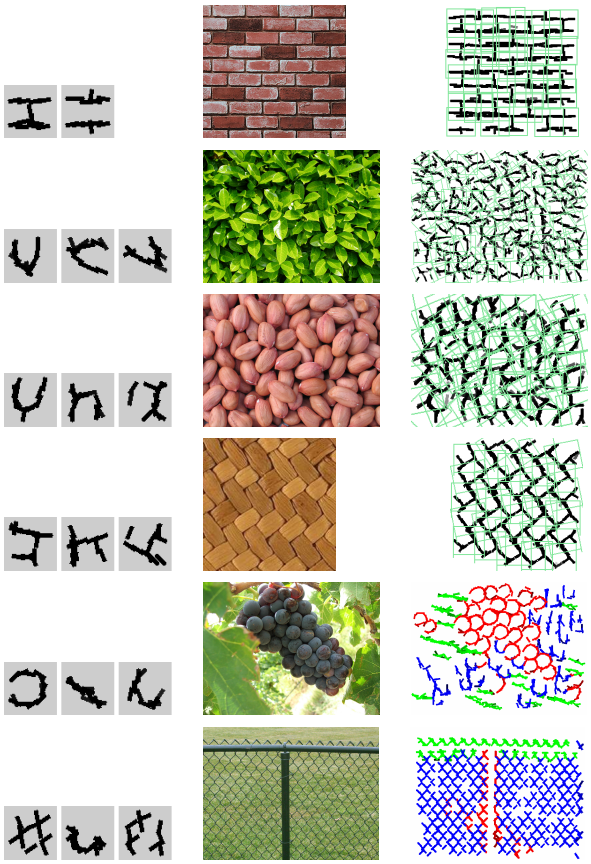

Figure 3. Learning codebooks of texons. For each texture image, we learn a codebook of 3 textons (for the wall example only 2 survived). The sizes of the images: bricks (291 by 271), leaves (509 by 382), peanuts (289 by 217), fabric (275 by 275), grapes (555 by 417), and fence (554 by 418). Each active basis template (i.e. entry in the texton codebooks) is of size 60 by 60 pixels, with no more than 12 selected wavelets. A small number of translated and deformed templates are used to pave each input image: bricks (43), leaves (94), peanuts (40), fabric (29), grapes (83), and fence (105). For the last two examples, we use different colors for different textons.

Allowing overlaps. In our current implementation, we allow quite some overlaps between the translated and deformed templates to avoid that situations where small pieces of images fall through the cracks and unpaved.

5.2. Learning codebooks of partons

We also apply the learning algorithm to object images *without assuming image alignment*. The learned codebooks of active basis templates correspond to parts of the objects as well as the backgrounds. These templates are recurring patterns across multiple images. We may call them partons, following the physicist Feynman. Figure 4 shows some examples.

Figure 4. Learning codebooks of partons. For each data set, we learn a codebook of 20 partons (for the flower example 19 survived). The number of training images: car (50), flower (9), teapot (61), and horse (50). Each active basis template (i.e. entry in the codebook) is of size 60 by 60 pixels, with no more than 12 selected wavelets. All images are resized to around 22,500 pixels.

5.3. Using codebooks of partons for classification

The learned codebooks of partons can be used for image classification. Specifically let $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$ be

Table 1. Classification accuracies on face and bicycle datasets.

dataset	method	codebook size	accuracy
Face	k-means+sift	50	84.15%
	k-means+sift	500	89.31%
	our approach	20	99.56%
Bicycle	k-means+sift	50	53.95%
	k-means+sift	500	76.69%
	our approach	50	81.23%

a codebook learned from positive training images, then for each testing image $\mathbf{I}$, we scan rotated versions of $\mathbf{B}^{(t)}$ over multiple resolutions of $\mathbf{I}$, and then take the global maximum of the template matching score as defined in section (3.5). We threshold the maximum score below at 0. This gives us T scores. We then feed these T scores to SVM [9] for classification.

We report experimental results on two commonly used datasets: Caltech-101 face [1] and Graz02 bicycle [2]. The Caltech-101 [2] contains 450 face images and 900 background images. Among them, we randomly select 225 face images for training, and all the other images for testing. The Graz02 dataset has 365 bicycle images and 380 background images. We train on 100 randomly selected bicycle images, and test on all the remaining bicycle and background images. For each data set, the accuracies are computed by averaging over 10 repetitions. We compared our method with k-means + sift [1, 4] under identical experiment settings. The latter is perhaps the most commonly used approach for image classification. Table 1 presents the experimental results. Our approach achieves a higher classification accuracy than k-means + sift with much smaller codebook size. Figure 5 gives the codebooks, each obtained from one repetition.

6. Conclusion

Although our work makes use of active basis model, our model is novel, and our learning algorithm is much more versatile and powerful than that used in the active basis model [10].

Reproducibility: Data and code can be found in the supplementary materials.

References

[1] G Csurka, C Dance, L Fan, J Willamowski, and C Bray. Visual categorization with bags of keypoints. *Workshop of ECCV*, 2004. 8

[1] downloaded from http://www.robots.ox.ac.uk/ vgg/data3.html

[2] downloaded from http: //www.emt.tugraz.at/ pinz/data/GRAZ_02/

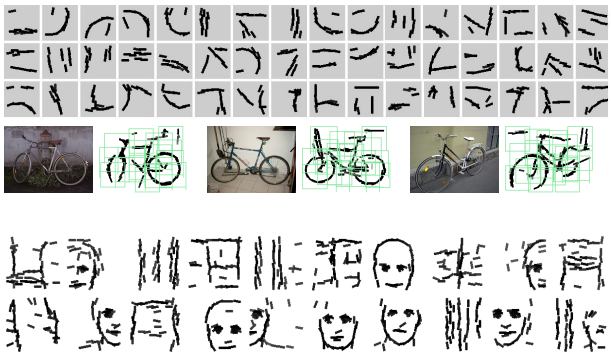

Figure 5. For bicycles we learn a codebook of 50 partons (48 survived, each parton has no more than 15 selected wavelets, size is 72 by 72 pixels) from 100 training images (each about 40,000 pixels). For faces, we learn a codebook of 20 templates (each template has no more than 40 selected wavelets, size is 120 by 120 pixels) from 225 images (each about 40,000 pixels).

[2] 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. 8

[3] JH Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987. 4

[4] D Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 8

[5] S Mallat and Z Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993. 3

[6] BA Olshausen and DJ Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996. 2

[7] BA Olshausen, P Sallee and MS Lewicki. Learning Sparse image codes using a wavelet pyramid architecture. *NIPS*, 13: 887-893, 2001. 2

[8] M Riesenhuber and T Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999. 3, 6

[9] VN Vapnik. *The nature of statistical learning theory*. 2000. 8

[10] YN Wu, Z Si, H Gong, and SC Zhu. Learning active basis model for object detection and recognition. *IJCV*, 90:198–235, 2010. 2, 3, 8

[11] L Zhu, C Lin, H Huang, Y Chen, and A Yuille. Unsupervised structure learning: hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *ECCV*, 2008. 2, 6

[12] SC Zhu, C Guo, Y Wang, and Z Xu. What are textons? *IJCV*, 62:121–143, 2005 2, 6

[13] SC Zhu and DB Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2:259–362, 2006. 2