Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?

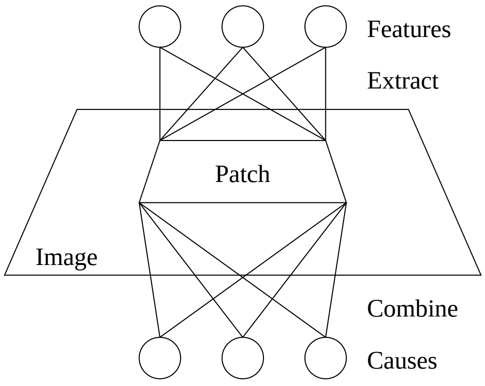
Bruno A. Olshausen **David J. Field**

December 1996

Contents of the Paper

1. Introduction (Why sparse and overcomplete?)
2. Image model (How to model nature?)
3. Probabilistic Framework
   (How to compare our model with nature?)
4. Learning Sparse Codes
   (How to find a good basis set for our model?)
5. Simulation (Tests by authors)
6. Results and Discussion

Image Model



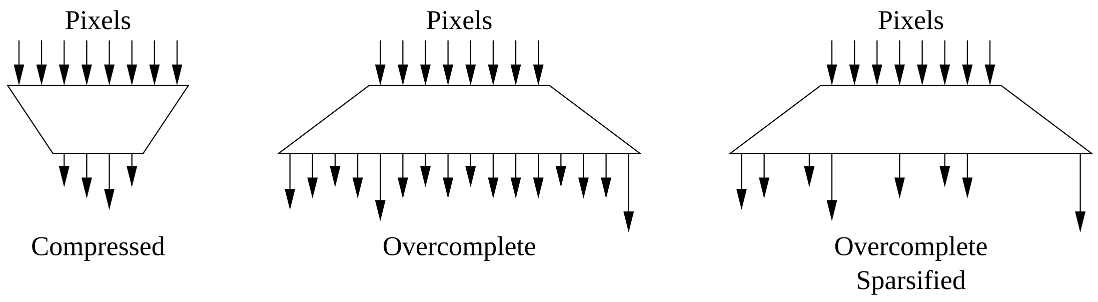
To analyze an image with filters ψ_i we can apply a linear transform to the image:

$$b_i = \sum_{\vec{x_j}} \psi_i(\vec{x_j}) I(\vec{x_j}). \quad (1)$$

Conversely, an image can be modeled as a combination of basis functions ϕ_i:

$$I(\vec{x}) = \sum_i a_i \phi_i(\vec{x}). \quad (2)$$

We are concerned with the case where the number of basis functions is greater than the number of inputs (pixels in an image).

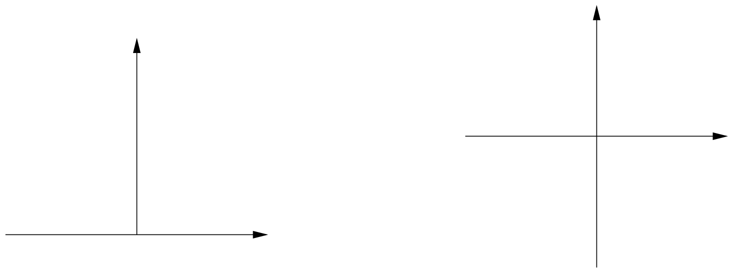


Probabilistic Framework

What kind of images arise from our model, when the basis functions are fixed, but the a_i are chosen randomly?

$$P(I|\phi) = \int P(I|a, \phi)P(a)da. \tag{3}$$

How do we choose the probablity distribution for the features a_i? We **assume** sparse structure.



We assume that features are independent: $P(a) = \prod_i P(a_i)$ and **choose** parameterization the distribution of the a_i:

$$P(a_i) = \frac{1}{Z_\beta} e^{-\beta S(a_i)} \tag{4}$$

where the funcion S determines the shape of the probability distribution and β its steepness. One possibility for S is

$$S = log(1 + x^2) \tag{5}$$

Learning Sparse Codes

We want to measure how well the distribution of the images generated by our model, $P(I|\phi)$, matches the actual distribution of natural images, $P^*(I)$. We use the Kullback-Leibler divergence.

$$KL = \int P^*(I) \log \frac{P^*(I)}{P(I|\phi)} dI. \tag{6}$$

Our goal is to find a set of bases, ϕ^*, which minimizes the KL. After an approximation we can write the problem in the following form:

$$\phi^* = \arg\min_{\phi} \langle \min_{a} E(I, a|\phi) \rangle, \tag{7}$$

where

$$E(I, a|\phi) = \sum_{\vec{x}} \Big[I(\vec{x}) - \sum_i a_i \phi_i(\vec{x}) \Big]^2 + \lambda \sum_i S(a_i). \tag{8}$$

Two Phases for Minimization

- In the inner loop we update the a_i until $E(I, a|\phi)$ is zero with respect to each a_i. For each image the a_i are determined from the equilibrium solution to the differential equation

$$\dot{a}_i = \sum_{\vec{x}} \phi_i(\vec{x}) r(\vec{x}) - \lambda S'(a_i) \tag{9}$$

where $r(\vec{x})$ is the residual image

$$r(\vec{x}) = I(\vec{x}) - \sum_i a_i \phi_i(\vec{x}). \tag{10}$$

The first term in equation 9 forces the ϕ_i to cover the image, and second term pushes most of the a_i towards zero.

- In the outer loop we minimize over the ϕ_i by simple gradient descent, which gives the learning rule

$$\Delta\phi_i(\vec{x}) = \eta \langle a_i r(\vec{x}) \rangle.$$

(Due to the earlier approximation of $P(I|\phi)$, the ϕ_i will grow without bound. So we have to adjust the L2 norm of each basis function ϕ_i.)

Simulation

Images

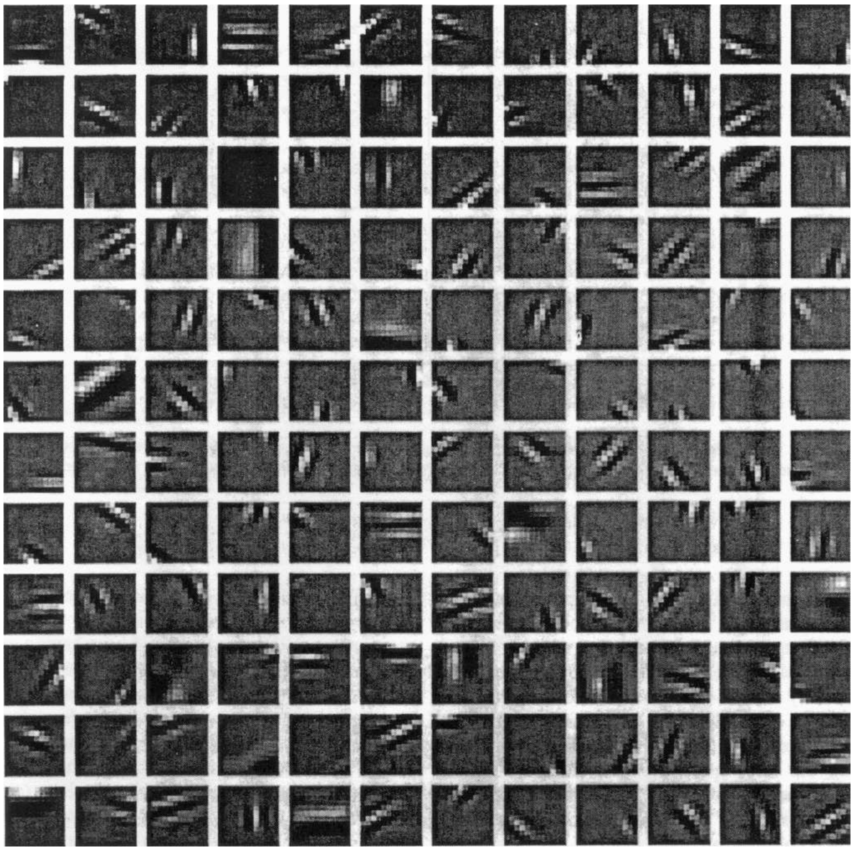
- Ten 512x512 images (trees, rocks, mountain scenes, etc.)
- Images were "sphered" to help gradient method.
- Corners of the 2D frequency domain were removed.
- Random 12x12 patches were extracted from the preprocessed images.

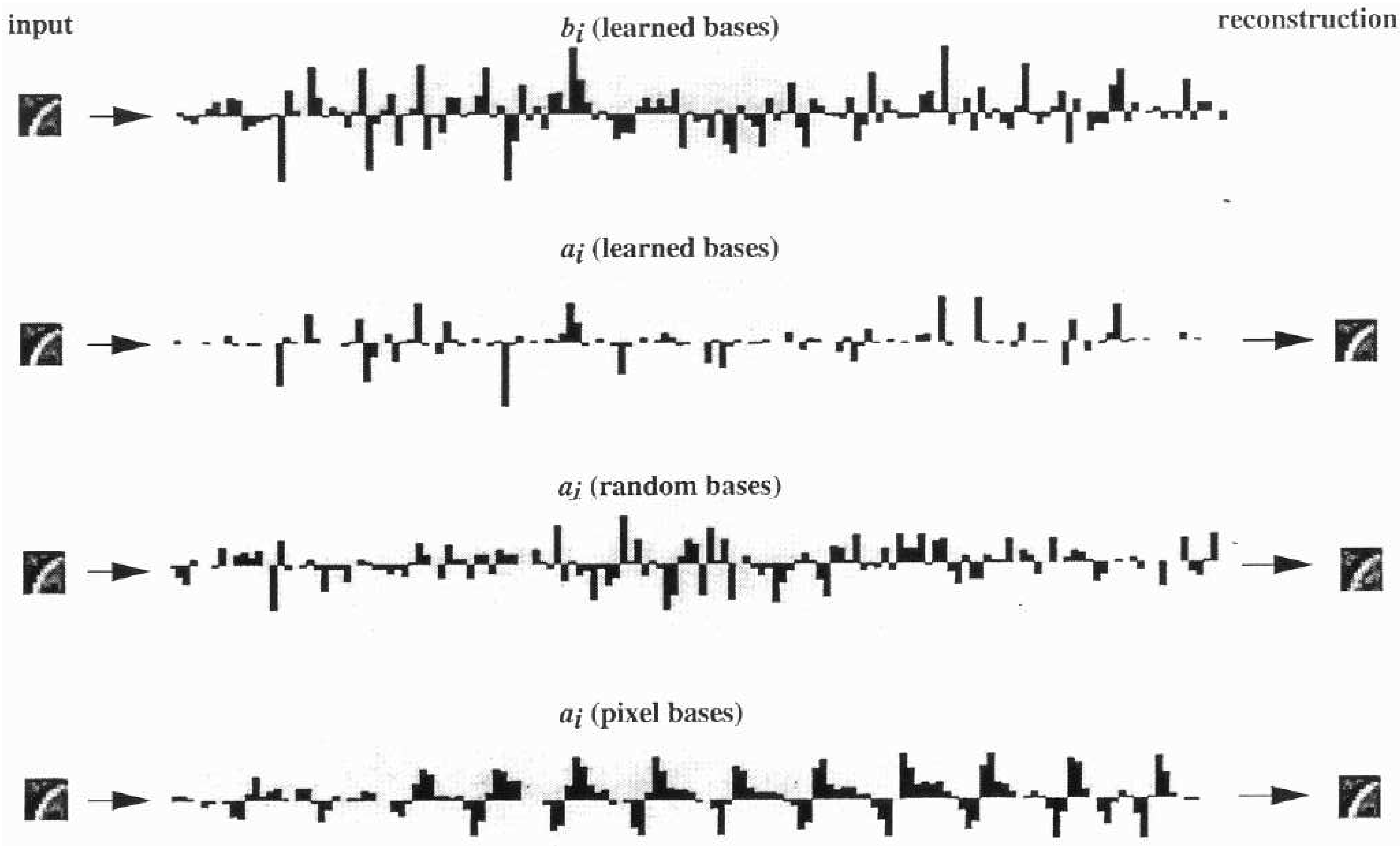
Learning

- The a_i were initialized: $a_i^0 = \sum_{\vec{x}} \phi_i(\vec{x}) I(\vec{x})$
- 144 basis functions were initialized to random values.
- Learning rate η was decreased gradually.
- A stable solution was reached after about 200000 image presentations.

Results

- The basis functions are well localized and oriented.
- The general form of the solution is robust.
- Preprocessing does not affect the overall qualitative appearance of the bases.
- Even if the number of basis functions equals the number of input pixels, the representation is effectively about 1.5-times overcomplete.




input
b_i (learned bases)
reconstruction
a_i (learned bases)
a_i (random bases)
a_i (pixel bases)

Conclusions

- A natural image is a combination of relatively small number of features selected from a large feature set.
- Sparsifying selects only the features which fit best in the image. Similar features which fit almost as well are pruned out.
- Because of we use sparse coding and overcomplete basis set, the input-output function is not purely linear.
- Using output for forming associations is easier because one does not have to consider units that are not necessary for representing a given image.

- However, because of sparse coding, small changes in the input may result in distinctly different patterns of neural activity.
- Note also that we assumed sparse structure in natural images. Sparse coding is not a general principle for finding statistically independent components in data.
- The model is based on a linear image model, but real causes of objects do not mix linearly (changes in size, rotations, etc.)