Noname manuscript No.
(will be inserted by the editor)

Learning Sparse FRAME Models for Natural Image Patterns

Jianwen Xie · Wenze Hu · Song-Chun Zhu · Ying Nian Wu

Received: date / Accepted: date

Abstract It is well known that natural images admit sparse representations by redundant dictionaries of basis functions such as Gabor-like wavelets. However, it is still an open question as to what the next layer of representational units above the layer of wavelets should be. We address this fundamental question by proposing a sparse FRAME (Filters, Random field, And Maximum Entropy) model for representing natural image patterns. Our sparse FRAME model is an inhomogeneous generalization of the original FRAME model. It is a non-stationary Markov random field model that reproduces the observed statistical properties of filter responses at a subset of selected locations, scales and orientations. Each sparse FRAME model is intended to represent an object pattern and can be considered a deformable template. The sparse FRAME model can be written as a shared sparse coding model, which motivates us to propose a two-stage algorithm for learning the model. The first stage selects the subset of wavelets from the dictionary by a shared matching pursuit algorithm. The second stage then estimates the parameters of the model given the selected wavelets. Our experiments show that the sparse FRAME models are capable of representing a wide variety of object patterns in natural images and that the learned models are useful for object classification.

Keywords Generative models · Markov random fields · Shared sparse coding

Department of Statistics
UCLA
E-mail: ywu@stat.ucla.edu

1 Introduction

1.1 Background and motivation

Sparsity underlies various types of data arising from different scientific disciplines. For natural images, the seminal work of Olshausen and Field (1996) [42] showed that natural image patches admit sparse linear representations by an over-complete or redundant dictionary of basis functions that resemble Gabor wavelets. The dictionary of basis functions can be learned from training image patches by minimizing the reconstruction error with a sparsity-inducing penalty as in the original work of Olshausen and Field. It can also be learned by pursuit-based algorithms such as K-SVD [3]. The learned dictionaries prove to be useful for tasks such as image recovery [5] [12] and image classification [61] [63].

During the past decade, a rich literature has been developed on learning dictionaries of wavelets for sparse coding. See, for example, the recent book by Elad [11] and the references therein. However, it remains an open question as to what the next layer of representational units above the layer of wavelets should be. The goal of this article is to address this fundamental question by proposing a class of statistical models for representing natural image patterns based on wavelets sparse coding. In this class of models, each model is composed of a subset of wavelets selected from the dictionary of wavelets. The model assumes that the image intensities are generated by a linear superposition of the selected wavelets, and the model implies a probability distribution on the coefficients of the selected wavelets.

1.2 Model and algorithm

One technical difficulty with modeling the coefficients of the selected wavelets explicitly is that it is difficult to specify the

multi-dimensional joint distribution of the coefficients, and yet it is unrealistic to assume that the coefficients are statistically independent. To get around the difficulty, we choose instead to model the responses of the image to the selected wavelets by adopting the mathematical form of the FRAME (Filters, Random field, And Maximum Entropy) model of Zhu, Wu, and Mumford (1997) [66] .

The FRAME model, which was originally proposed for stochastic texture patterns, is a spatially stationary or homogeneous Markov random field model. Furthermore, it is the maximum entropy distribution that reproduces the observed marginal histograms of responses from a bank of filters, where for each filter tuned to a specific scale and orientation, the marginal histogram is spatially pooled over all the pixels in the image domain.

By modifying the original FRAME model, we propose a sparse FRAME model for representing natural image patterns. It is a generative model with a well-defined probability distribution on the image intensities. Unlike the original FRAME model for texture patterns, each sparse FRAME model is intended to model an object pattern, and can be considered a deformable template for this pattern. It is spatially non-stationary or inhomogeneous, and it is the maximum entropy distribution that reproduces statistical properties of filter responses at a subset of selected locations, scales and orientations.

The sparse FRAME model can be written as a shared sparse coding model, where the observed images are represented by a commonly shared subset of wavelets at selected locations, scales and orientations, subject to local perturbations to account for shape deformations. The sparse FRAME model implicitly assumes a joint probability distribution on the coefficients of the selected wavelets.

We then propose a two-stage algorithm to learn the sparse FRAME model from roughly aligned image patches. The first stage selects a subset of wavelets to simultaneously reconstruct all the observed images, while allowing the selected wavelets to perturb their locations and orientations to represent each individual image. The second stage of the algorithm then estimates the parameters of the model given the selected wavelets. This stage implicitly estimates the probability distribution of the coefficients of the subset of selected wavelets. The computation of the second stage can be accomplished by stochastic gradient ascent [62], which seeks to reproduce the observed statistical properties of the responses of the selected wavelets. Our experiments show that the learned model or template can synthesize realistic images and can be used to detect similar object patterns in the testing images. The learning algorithm can also be used for clustering images of different patterns.

The above two-stage algorithm can learn a single sparse FRAME model from a training set of aligned images. It can also be employed to learn a codebook of sparse FRAME models or templates from non-aligned training images, so that each image can be represented by a small number of spatially translated, rotated and scaled versions of the templates selected from the learned codebook. We use an unsupervised learning algorithm that iterates the following two steps: (1) Image encoding: Given the current codebook, select templates to encode the training images using a template matching pursuit algorithm. (2) Codebook re-learning: Given the encoding of the training images, re-learn the templates from the training image patches by the two-stage learning algorithm. Our experiments show that it is possible to learn codebooks of sparse FRAME models and the learned models are useful for image classification.

In this paper, we assume that the bank of filters is given. For example, the filter bank contains Gabor filters and Difference of Gaussian (DoG) filters as in the original FRAME model. In other words, we assume that there already exists a dictionary of wavelets that gives sparse representations of the observed images. Presumably this dictionary can itself be learned if there are enough training data. We shall not pursue this issue in this paper, and shall focus on learning the generative models based on the given dictionary of wavelets.

1.3 Related work

The two stages of the learning algorithm for training the sparse FRAME model naturally connect two major frameworks in image representation and modeling, namely the sparse coding framework with its root in harmonic analysis and the Markov random field framework with its root in statistical physics. There have been vast literatures on both themes of research. In the following, we shall review and compare with some of the papers that are most relevant to our work.

Markov random field models. Models in this class are also called energy-based models [53] [1], exponential family models, and Gibbs distributions depending on the context. Examples include the FRAME model [66] as well as its inhomogeneous extension for face shape data [33], field of experts [47], product of experts [26], product of t model [58], restricted Boltzmann machine [52] [27] and its many recent generalizations such as those found in [45] and the references therein. A Markov random field model is defined by an energy function and may involve latent variables or hidden units. If the latent variables are conditionally independent given the observed data or visible units, the latent variables can be integrated out in closed form, resulting in a marginal energy-based model. These models usually assume fixed energy functions and do not involve explicit feature selection. In addition, these models seek to approximate the probability distributions of the training images but do not attempt to reconstruct individual training images. Compared to these models, the sparse FRAME model performs feature

selection via a linear additive model and can reconstruct the training images.

Sparse coding models. The sparse FRAME model selects the wavelets via a shared sparse coding scheme. Such a scheme has been studied in harmonic analysis and signal processing [7] [55] under the name of simultaneous sparse coding, and in statistics and machine learning [41] [34] under the names of multi-task learning and support union recovery. These methods seek to reconstruct the observed signals but do not attempt to approximate the probability distributions of these signals. In contrast, the sparse FRAME model defines an explicit probability distribution on image intensities and can synthesize new images by sampling from this distribution.

There has also been work on learning dictionaries that are sparse combinations of wavelets from a base dictionary, such as [48], where the coefficients of sparse linear combinations are fixed. In our model, the coefficients are allowed to vary according to a certain probability distribution. In addition, the wavelets are also allowed to perturb their locations and orientations. Our work is also related to subspace clustering [2], where each cluster is spanned by a subset of wavelets or basis functions. In subspace clustering, the distributions of the coefficients of the basis functions that span the subspaces are not modeled. In our work, we seek to model the coefficients or responses of the selected basis functions.

The proposed model (and the original FRAME model) is related to "analysis priors" developed in the last few years by the sparse modeling community [38] [13]. Our learning algorithm can be viewed as a principled statistical method to learn the parameters or weights of the analysis prior models via maximum likelihood. Here our focus is on statistical modeling and random sampling, whereas the focus of the analysis priors is on optimization and reconstruction tasks.

Deep learning. Our work is not exactly within the domain of deep learning, but is closely related to it. In particular, we try to understand the layer of representational units or nodes above the layer of sparse coding wavelets. Our proposal is that each node is a sparse FRAME model, which is selectively and sparsely connected to a subset of wavelets selected for this model. The connection weights are the parameters of the model. The sparse connections are obtained by seeking the shared sparse coding of a collection of image patches of similar patterns. The advantage of seeking the shared sparse coding is that the time-consuming explaining-away computation in sparse coding can be memorized in the learning stage by sparse connections, so that explaining-away sparsification does not need to be recalculated on-line in the inference stage. In other words, we believe that "sparse connectivities = shared sparse activities" or "sparse wiring = shared sparse coding". In contrast, the current methods of deep learning are mostly based on stacking restricted Boltzmann machines (RBM) [27] [32] or auto-encoders [4]. They do not pursue explicit sparse representations and sparse connections. Our work is also related to the deconvolution network of [63]. It appears that in the deconvolution model, given the values of the top layer units, the values of the units at lower layers are fixed by the learned weights, and are not allowed to vary. In our model, we allow the coefficients to vary according to the learned probability distribution.

Our method can be extended to learning hierarchical models. After learning one layer of sparse FRAME models, we can treat these models as re-usable parts, and continue to compose them into higher layers of sparse FRAME models.

Compositional models. The sparse FRAME model is a special case of compositional models advocated by S. Geman et al. for vision [22]. In particular, it follows the And-Or grammar studied by Zhu and Mumford [65], where the composition of the wavelets forms an And-node, and the perturbation of each selected wavelet and the variation of its coefficient form an Or-node. Our model is also related to [64] [18] , which are about compositions of edgelets but which are not based on explicit generative models as in our work.

Compared to our own previous work, this paper can be considered a fusion of the original FRAME model [66] and the active basis model [59] [29]. While the active basis model focuses on the "sketching" aspect, this paper adds the "painting" aspect. In order to avoid MCMC computation in learning, the active basis model makes the simplifying assumptions that the selected wavelets are orthogonal and their coefficients are statistically independent. In this paper, we do not make such simplifying assumptions, and thus our model is more rigorously defined and is capable of synthesizing realistic image patterns. This paper is an expanded version of our conference paper [60].

1.4 Contributions

The following are the main contributions of this paper. (1) We propose an inhomogeneous dense FRAME model for object patterns, and we show that it can model a wide variety of objects in natural scenes. (2) We propose a sparse FRAME model and connect it to the shared sparse coding model. We then propose a two-stage algorithm for learning the sparse FRAME model. (3) We show that it is possible to learn codebooks of sparse FRAME models from non-aligned and unannotated images.

2 Inhomogeneous FRAME model

This section presents a dense version of the inhomogeneous FRAME model to lay the foundation for the next section which will focus on the sparsified version.

Fig. 1: The inhomogeneous FRAME is a generative model that seeks to represent and generate object patterns as shown above.

2.1 Model and learning algorithm

Notation. We start by modeling roughly aligned images of object patterns from the same category, such as the images in Figure 1. Let $\{\mathbf{I}_m, m = 1, ..., M\}$ be a set of training images defined on an image domain $\mathcal{D}$. We use the notation $B_{x,s,\alpha}$ to denote a basis function such as a Gabor wavelet centered at pixel x (which is a two-dimensional vector), and tuned to scale s and orientation α. $B_{x,s,\alpha}$ is also an image on $\mathcal{D}$, although it is non-zero only within a local range. We assume that $B_{x,s,\alpha}$ are translated, dilated and rotated versions of each other. We assume that s and α take values within a finite and properly discretized range. The inner product $\langle \mathbf{I}, B_{x,s,\alpha} \rangle$ can be considered the filter response of $\mathbf{I}$ to a filter of scale s and orientation α at pixel x. We assume that $B_{x,s,\alpha}$ are all normalized to have unit ℓ_2 norm.

Model. The inhomogeneous FRAME model is a probability distribution defined on $\mathbf{I}$,

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}(\langle \mathbf{I}, B_{x,s,\alpha} \rangle) \right) q(\mathbf{I}), \quad (1)$$

where $q(\mathbf{I})$ is a known reference distribution or a null model, $\lambda_{x,s,\alpha}()$ are one-dimensional functions that depend on (x, s, α), $\lambda = \{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$, and

$$Z(\lambda) = \int \exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}(\langle \mathbf{I}, B_{x,s,\alpha} \rangle) \right) q(\mathbf{I}) d\mathbf{I} \quad (2)$$

$$= \mathrm{E}_q \left[\exp\left(\sum_{x,s,\alpha} \lambda_{x,s,\alpha}(\langle \mathbf{I}, B_{x,s,\alpha} \rangle) \right) \right] \quad (3)$$

is the normalizing constant, where the notation E_q means the expectation with respect to the probability distribution q.

In the original FRAME model for stochastic textures [66], $\lambda_{x,s,\alpha}()$ is assumed to be independent of x (but dependent of s and α, which index the scale and orientation of the filter). So the model is spatially stationary. For modeling object patterns that are not spatially stationary, $\lambda_{x,s,\alpha}()$ must depend on x, in addition to s and α.

In the original homogeneous FRAME, the potential functions $\lambda_{s,\alpha}()$ (we drop the subscript x due to stationarity) are estimated non-parametrically as step functions. In the inhomogeneous FRAME, we have to estimate $\lambda_{x,s,\alpha}()$ for each individual x. With small data sets, we may not afford estimating $\lambda_{x,s,\alpha}()$ non-parametrically. We therefore decide to parametrize

$$\lambda_{x,s,\alpha}(r) = \lambda_{x,s,\alpha}|r|, \quad (4)$$

where $r = \langle \mathbf{I}, B_{x,s,\alpha} \rangle$, and with slight abuse of notation, $\lambda_{x,s,\alpha}$ on the right hand side of the above equation becomes a constant (instead of a function as on the left hand side). The parametrization (4) is inspired by the Laplacian distribution that can account for heavy tails in the distributions of filter responses. It is possible to replace the function $|r|$ by other classes of parametrized functions to encourage heavy tails of the responses, and we shall investigate this issue in future work.

In many Markov random field models including the original FRAME model, the reference measure $q(\mathbf{I})$ is simply the uniform measure. In our work, we assume $q(\mathbf{I})$ to be the Gaussian white noise model, under which the image intensities follow independent $\mathrm{N}(0, \sigma^2)$ distributions. So

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{|\mathcal{D}|/2}} \exp\left(-\frac{1}{2\sigma^2} \sum_x \mathbf{I}(x)^2 \right), \quad (5)$$

where $|\mathcal{D}|$ is the number of pixels in the image domain. $q(\mathbf{I})$ itself is a maximum entropy model relative to a uniform measure, and it reproduces the marginal mean and variance of the image intensities. In our work, we normalize the observed images to have marginal mean 0 and variance 1, so we choose $\sigma^2 = 1$. This $q(\mathbf{I})$ can be considered an initial model or a model of the background residual image with the foreground object removed. As a result, $p(\mathbf{I}; \lambda)$ in equation (1) can be written as an exponential family model relative to the uniform measure.

Maximum likelihood learning. The inhomogeneous version of the FRAME model is a special case of the exponential family model, and the parameter $\lambda = (\lambda_{x,s,\alpha}, \forall x, s, \alpha)$ can be estimated from the training images $\{\mathbf{I}_m, m = 1, ..., M\}$ by maximum likelihood. The log-likelihood function is

$$L(\lambda) = \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{I}_m; \lambda) \quad (6)$$

$$= \frac{1}{M} \sum_{m=1}^{M} \sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle| - \log Z(\lambda) \quad (7)$$

$$+ \frac{1}{M} \sum_{m=1}^{M} \log q(\mathbf{I}_m).$$

The maximization of $L(\lambda)$ can be accomplished by gradient ascent. The gradient is

$$\frac{\partial L(\lambda)}{\partial \lambda_{x,s,\alpha}} = \frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle| - \mathrm{E}_{p(\mathbf{I};\lambda)} \left[|\langle \mathbf{I}, B_{x,s,\alpha} \rangle| \right], \quad \forall x, s, \alpha, \quad (8)$$

where $\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ is the expectation of $|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|$ with $\mathbf{I}$ following the distribution $p(\mathbf{I};\lambda)$. $\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ is the derivative of $\log Z(\lambda)$.

The gradient ascent algorithm then becomes

$$\lambda^{(t+1)}_{x,s,\alpha} = \lambda^{(t)}_{x,s,\alpha} + \gamma_t \left(\frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|] \right), \tag{9}$$

where γ_t is the step size. The analytic form of the expectation under the current model at step t, $\mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$, is not available, so we approximate it from a sample set of synthesized images $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ generated from $p(\mathbf{I};\lambda^{(t)})$:

$$\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|] \approx \frac{1}{\tilde{M}}\sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha}\rangle|. \tag{10}$$

The synthesized images $\{\tilde{\mathbf{I}}_m\}$ can be sampled from $p(\mathbf{I};\lambda^{(t)})$ by Hamiltonian Monte Carlo (HMC) [40]. Unlike the Gibbs sampler [21], HMC makes use of the gradient of the energy function, and it is particularly natural for our model. The computation of HMC involves a bottom-up convolution step followed by a top-down convolution step. Both steps can be efficiently implemented in Matlab by GPU. With HMC and warm start, $\{\tilde{\mathbf{I}}_m\}$ are produced by $\tilde{M}$ parallel chains. All the synthesized images presented in the figures of this paper are generated by the HMC algorithm along the learning process. More details about simulation by the HMC algorithm are presented in Section 8.1 in the appendix.

With $\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ approximated according to (10), we arrive at the stochastic gradient algorithm analyzed by Younes (1999) [62] :

$$\lambda^{(t+1)}_{x,s,\alpha} = \lambda^{(t)}_{x,s,\alpha} + \gamma_t \left(\frac{1}{M}\sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha}\rangle| - \frac{1}{\tilde{M}}\sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha}\rangle| \right). \tag{11}$$

This is the algorithm we use for maximum likelihood estimation of λ.

Computing normalizing constants. Thanks to HMC, we can simulate from $p(\mathbf{I};\lambda)$ without knowing its normalizing constant, thus estimating λ by MLE. Nevertheless, computing the normalizing constant $Z(\lambda)$ is still required in situations such as fitting a mixture model or learning a codebook of models. The ratio of the normalizing constants at two consecutive steps is

$$\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})} = \mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}\left[\exp\Big(\sum_{x,s,\alpha} (\lambda^{(t+1)}_{x,s,\alpha} - \lambda^{(t)}_{x,s,\alpha}) \times |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| \Big) \right] \tag{12}$$

which can be approximated by averaging over the sampled images $\{\tilde{\mathbf{I}}_m\}$ as an application of importance sampling [20]:

$$\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})} \approx \frac{1}{\tilde{M}}\sum_{m=1}^{\tilde{M}} \left[\exp\Big(\sum_{x,s,\alpha} (\lambda^{(t+1)}_{x,s,\alpha} - \lambda^{(t)}_{x,s,\alpha}) \times |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha}\rangle| \Big) \right]. \tag{13}$$

Starting from $\lambda^{(0)} = 0$ and $\log Z(\lambda^{(0)}) = 0$, we can compute $\log Z(\lambda^{(t)})$ along the learning process by iteratively updating its value as follows:

$$\log Z(\lambda^{(t+1)}) = \log Z(\lambda^{(t)}) + \log \frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}. \tag{14}$$

The calculation of Z is based on running parallel Markov chains for a sequence of distributions $p(\mathbf{I};\lambda^{(t)})$. The setting is similar to annealed importance sampling [39] and bridge sampling [20]. We shall explore these methods in future work.

2.2 Summary of the learning algorithm

Pseudocode of the algorithm for learning the inhomogeneous FRAME model is shown in Algorithm 1. The algorithm stops when the gradient of the log-likelihood is close to 0, i.e., when the statistics of the synthesized images closely match those of the observed images. Figure 2 displays the synthesized images $\{\tilde{\mathbf{I}}_m\}$ generated by the models learned from training images shown in Figure 1 (a separate model is learned from each training set). Figure 3 illustrates the learning process by showing the synthesized images with λ being updated by the algorithm. The synthesized image starts from Gaussian white noises sampled from $q(\mathbf{I})$, then gradually gets similar to the observed images in the overall shape and appearance.

The computational complexity of Algorithm 1 is of the order $O(U \times \tilde{M} \times L \times K \times H_B \times W_B)$ with U being the number of updating steps for λ, $\tilde{M}$ the number of synthesized images, L the number of leapfrog steps in HMC, K the number of filters, and H_B and W_B the average window sizes (height and width) of the filters. As to the actual running time, for the cat example, each iteration of a single chain takes about 2 seconds on a current PC, with $L = 30$, $K = 240100$, $H_B = 12$, and $W_B = 12$.

Experiment 1: Learning dense FRAME. Figure 4 displays some images generated by the dense models learned from roughly aligned training images. We run a single chain in the learning process, i.e., $\tilde{M} = 1$ in this experiment. The learned models can generate a wide variety of natural image patterns. Typical sizes of the images are 70×70.

In the appendix, Section 8.2 gives a justification of the inhomogeneous FRAME model by the maximum entropy principle.

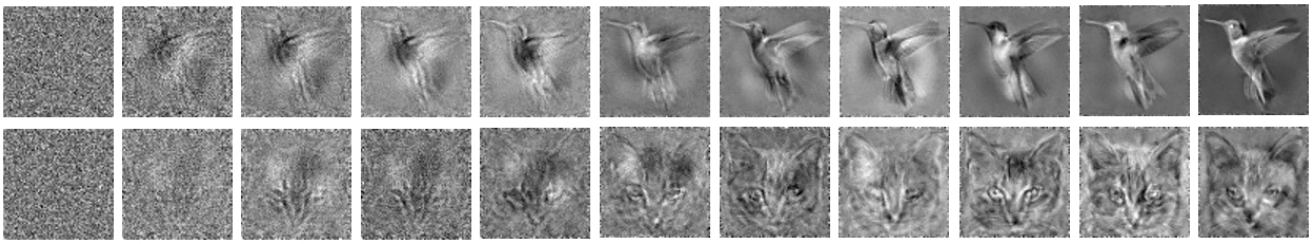

Fig. 3: Learning sequence by inhomogeneous FRAME. The sizes of the images are 70×70. A separate model is learned from each training set shown in Fig 1 (Top: Hummingbird. The number of training images is 5 as shown in Fig. 1. Bottom: Cat. The number of training images is 12, with 6 of them shown in Fig. 1.) Synthesized images generated in iterations $t = 1, 4, 7, 10, 13, 20, 50, 100, 200, 300, 400$, and 500.

Fig. 2: Synthesized images generated by the inhomogeneous FRAME models learned separately from a training set of 5 hummingbird images and another training set of 12 cat images. Some of the images are displayed in Fig. 1. The sizes of the images are 70×70.

Algorithm 1 Learning algorithm for dense FRAME

Input:
training images $\{\mathbf{I}_m, m = 1, ..., M\}$
Output:
$\lambda = \{\lambda_{x,s,\alpha}, \forall x, s, \alpha\}$ and $\log Z(\lambda)$

1: Create a filter bank $\{B_{x,s,\alpha}, \forall x, s, \alpha\}$
2: Initialize $\lambda^{(0)}_{x,s,\alpha} \leftarrow 0, \forall x, s, \alpha$.
3: Calculate observed statistics:
$H^{obs}_{x,s,\alpha} \leftarrow \frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle|, \forall x, s, \alpha$.
4: Initialize synthesized images $\tilde{\mathbf{I}}_m$ as Gaussian white noise images
5: Initialize $\log Z(\lambda^{(0)}) \leftarrow 0$
6: Let $t \leftarrow 0$
7: **repeat**
8: Generate $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ from $p(\mathbf{I}; \lambda^{(t)})$ by HMC
9: Calculate synthesized statistics:
$H^{syn}_{x,s,\alpha} \leftarrow \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle|, \forall x, s, \alpha$.
10: Update $\lambda^{(t+1)}_{x,s,\alpha} \leftarrow \lambda^{(t)}_{x,s,\alpha} + \gamma_t (H^{obs}_{x,s,\alpha} - H^{syn}_{x,s,\alpha}), \forall x, s, \alpha$.
11: Compute Z ratio $\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$ by Eq. (13)
12: Update $\log Z(\lambda^{(t+1)}) \leftarrow \log Z(\lambda^{(t)}) + \log \frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$
13: Let $t \leftarrow t + 1$
14: **until** $\sum_{x,s,\alpha} |H^{obs}_{x,s,\alpha} - H^{syn}_{x,s,\alpha}| \leq \epsilon$

3 Sparse FRAME model

Sparsification. In model (1), the (x, s, α) in $\sum_{x,s,\alpha}$ is over all the pixels x and all the scales s and orientations α. We call such a model the dense FRAME. It is possible to sparsify the model by selecting only a small set of (x, s, α) so that $\sum_{x,s,\alpha}$ is restricted to this selected subset. More explicitly, we can write the sparsified model as

$$p(\mathbf{I}; \mathbf{B}, \lambda) = \frac{1}{Z(\lambda)} \exp\left(\sum_{i=1}^{n} \lambda_i |\langle \mathbf{I}, B_{x_i,s_i,\alpha_i} \rangle| \right) q(\mathbf{I}), \quad (15)$$

where $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ are the selected basis functions, and $\lambda = (\lambda_i, i = 1, ..., n)$ collects the parameters. Given the selected basis functions $\mathbf{B}$, the model can still be trained by maximum likelihood as in the previous section, and properties such as maximum entropy still hold. The following are the reasons why a sparsified model is desirable. (1) It makes the computation faster. (2) It leads to more reliable parameter estimates because it involves a much smaller number of parameters. Estimation efficiency or accuracy is an important aspect of statistical modeling. (3) The MCMC sampling may converge faster if the selected basis functions are not heavily correlated. (4) It is connected to the linear additive sparse coding model for image reconstruction. (5) It allows the selected basis functions to perturb their locations and orientations to account for shape deformations.

Deformation. To be more specific about the above point (5), we may treat $p(\mathbf{I}; \mathbf{B}, \lambda)$ as a deformable template, so that when it is fitted to each training image $\mathbf{I}_m$, we may allow the basis functions in $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ to perturb their locations and orientations so that $\mathbf{B}$ is deformed to $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i},s_i,\alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$, where $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ are the perturbations of the location and orientation of the i-th basis function B_{x_i,s_i,α_i}. 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 along the normal direction of the Gabor wavelet, and $\Delta\alpha_{m,i} \in \{-1, 0, 1\} \times \pi/16$). When we fit the model $p(\mathbf{I}; \mathbf{B}, \lambda)$ to $\mathbf{I}_m$, we model $\mathbf{I}_m$ by $p(\mathbf{I}_m; \mathbf{B}_m, \lambda)$, in which B_{x_i,s_i,α_i} in (15) is changed to $B_{x_i+\Delta x_{m,i},s_i,\alpha_i+\Delta\alpha_{m,i}}$.

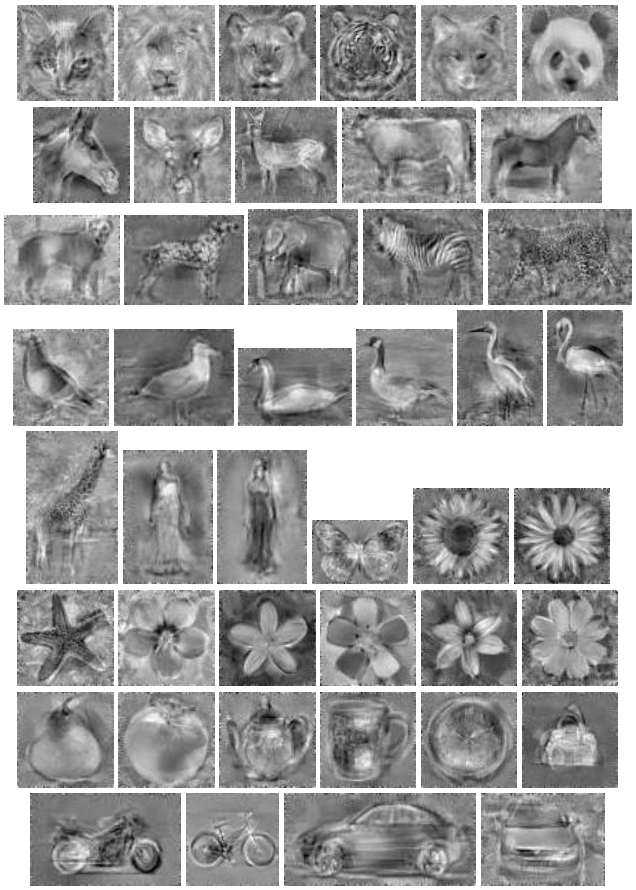

Fig. 4: Synthesis by dense FRAME. Images generated by the dense FRAME models learned from different categories of objects. The training images are collected from the internet and are cropped so that the training images for each category are roughly aligned. Typical number of training images for each category is around 10.

3.1 Shared sparse coding

We can select B_{x_i,s_i,α_i} or (x_i, s_i, α_i) in model (15) sequentially using a procedure like projection pursuit [19] or filter pursuit [66], but the computational speed of such a sequential procedure can be slow. In this article, we choose to employ a different strategy by exploring the connection between sparse FRAME and shared sparse coding.

For simplicity, let us temporarily ignore the issue of deformation. For the sparse model in equation (15), the number of selected basis functions n is always much smaller than the number of pixels $|\mathcal{D}|$. We can then project each $\mathbf{I}_m \sim p(\mathbf{I}; \lambda)$ onto the subspace spanned by the selected basis functions $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$, so that

$$\mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{x_i,s_i,\alpha_i} + \epsilon_m, \tag{16}$$

where $c_{m,i}$ are the least squares reconstruction coefficients of the linear projection, and ϵ_m is the resulting residual image that resides in the $|\mathcal{D}| - n$ dimensional residual subspace that is orthogonal to the subspace spanned by $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$. Equation (16) is a shared space coding model, where the small set of basis functions $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ is shared by the training images $\{\mathbf{I}_m, m = 1, ..., M\}$.

With the Gaussian white noise background model $q(\mathbf{I})$ where each $\mathbf{I}(x) \sim \mathrm{N}(0, \sigma^2)$ independently, the sparse model (15) implies that $C_m = (c_{m,i}, i = 1, ..., n)$ follows a certain distribution $p_C(C; \lambda)$, ϵ_m is the projection of the Gaussian white noise image onto the $|\mathcal{D}| - n$ residual subspace, and C_m and ϵ_m are independent of each other. The log-likelihood of $\mathbf{I}_m$ can be decomposed into the log-likelihood of C_m and the log-likelihood of the projected Gaussian white noise ϵ_m. While the former depends on λ, the latter only depends on the squared norm of the residual image $\|\epsilon_m\|^2 = \|\mathbf{I}_m - \sum_{i=1}^{n} c_{m,i} B_{x_i,s_i,\alpha_i}\|^2$.

The above consideration suggests a two-stage learning algorithm for fitting the sparse FRAME model. In the first stage, we selected $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ by minimizing the overall least squares reconstruction error $\sum_{m=1}^{M} \|\epsilon_m\|^2$. In the second stage, we then estimate λ given the selected $\mathbf{B}$.

In the appendix, Section 8.3 gives a more detailed explanation of the connections between the sparse FRAME model and the shared sparse coding model. In particular, it shows that the sparse FRAME model is equivalent to the shared sparse coding model with an implied joint distribution on the coefficients of the selected basis functions.

Now let us consider the issue of deformation. Since model (15) is deformable, we can also make the sparse coding model (16) deformable by allowing the shared basis functions to perturb their locations and orientations to account for the shape deformation in each image. This leads to the deformable shared sparse coding first proposed in our previous work on active basis [59]

$$\mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i},s_i,\alpha_i+\Delta\alpha_{m,i}} + \epsilon_m, \tag{17}$$

where $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ are the perturbations of the location and orientation of the i-th basis function.

3.2 The two-stage learning algorithm

This subsection describes the learning algorithm for training the sparse FRAME model, which consists of two stages. (1) Selecting $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ by shared sparse coding. (2) Estimating $\lambda = (\lambda_i, i = 1, ..., n)$ given the selected $\mathbf{B}$.

Stage 1: Deformable shared sparse coding. For training images $\{\mathbf{I}_m, m = 1, ..., M\}$, we select the basis functions

Algorithm 2 Stage 1: Deformable shared matching pursuit

Input:

training images $\{\mathbf{I}_m, m = 1, ..., M\}$

Output:

selected basis functions $\mathbf{B} = \{B_{x_i,s_i,\alpha_i}, i = 1, ..., n\}$

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

2: Let $i \leftarrow i + 1$. Then we select

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

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

3: For each m, given (x_i, s_i, α_i), infer the perturbations in locations and orientations by retrieving the arg-max in the local max pooling of step 2:

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

Let the coefficient

$$c_{m,i} \leftarrow \langle \epsilon_m, B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}}\rangle,$$

and update the residual image by explaining away:

$$\epsilon_m \leftarrow \epsilon_m - c_{m,i} B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}}.$$

4: Stop if $i = n$, else go back to step 2.

$\{B_{x_i s_i,\alpha_i}, i = 1, ..., n\}$ by minimizing

$$\sum_{m=1}^{M} \|\mathbf{I}_m - \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}}\|^2. \quad (18)$$

The minimization can be accomplished by the shared matching pursuit algorithm [36] [59] that selects basis functions to encode multiple images simultaneously, while inferring local perturbations by local max pooling [46]. The algorithm is presented in Algorithm 2.

We can replace the matching pursuit component in the above algorithm by the orthogonal matching pursuit [43], which is more computationally expensive. We can also replace the matching pursuit component by penalized least squares such as basis pursuit [6] or Lasso [54], or more precisely using a penalty such as the ℓ_1/ℓ_2 group norm [41]. The computation can be much more expensive than shared matching pursuit.

Simultaneous sparse approximation of multiple signals has been studied in the harmonic analysis and machine learning literature [55] [41]. However, perturbations of the selected basis functions are not considered in these papers. Such a deformable shared matching pursuit algorithm was first proposed by [59], but it implemented a modified version that enforces approximated orthogonality of the selected basis functions.

Stage 2: Sparse FRAME as deformable template. After selecting $\mathbf{B} = \{B_{x_i,s_i,\alpha_i}, i = 1, ..., n\}$, we can then model

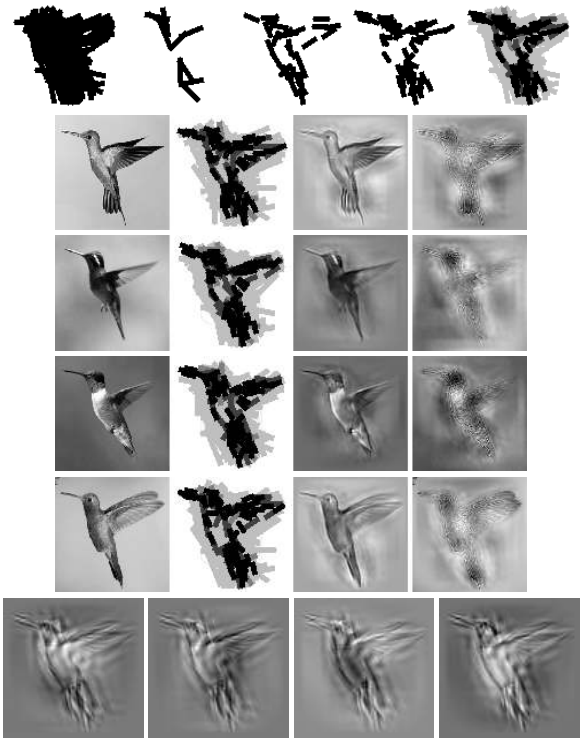

Fig. 5: Reconstruction and synthesis by sparse FRAME model (hummingbirds). The number of selected wavelets is 300. The first row contains symbolic sketches of selected Gabor wavelets at different scales, where each Gabor wavelet is illustrated by a bar. The first 4 sketches correspond to 4 different scales. The last one is the superposition of the 4 scales, where smaller scales appear darker. The next 4 rows display examples of the training images, the deformed sketches, the reconstructed images, and the residual images. The last row displays examples of synthesized images generated by the learned model. The number of training images is 5 as shown in Fig. 1. The sizes of images are scaled to 100 $\times$ 100.

$\{\mathbf{I}_m\}$ by the sparse FRAME model (15), by estimating λ via MLE. $p(\mathbf{I}; \mathbf{B}, \lambda)$ in (15) now serves as the deformable template in that the log-likelihood of $\mathbf{I}_m$ is

$$L(\mathbf{I}_m|\mathbf{B}, \lambda) = \sum_{i=1}^{n} \lambda_i \max_{\Delta x,\Delta\alpha} |\langle \mathbf{I}_m, B_{x_i+\Delta x, s_i, \alpha_i+\Delta\alpha}\rangle| - \log Z(\lambda), \quad (19)$$

which serves as the template matching score. We allow each selected B_{x_i,s_i,α_i} to perturb its location and orientation to account for shape deformation, where the perturbation is inferred by the local max pooling in Algorithm 2.

In the learning algorithm, again, let $\lambda^{(t)}$ be the current estimate of λ, and let $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ be the synthesized images drawn from $p(\mathbf{I}; \lambda^{(t)})$ by $\tilde{M}$ parallel chains.

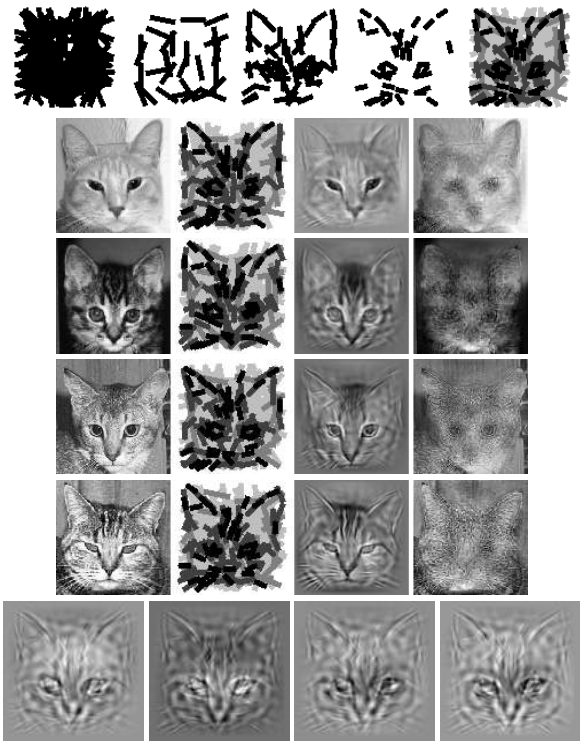

Fig. 6: Reconstruction and synthesis (cats). See caption of Fig. 5. The number of training images is 12, with 6 of them shown in Fig. 1. The sizes of images are scaled to 100 × 100. The number of selected wavelets is 300.

Then we update λ by

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \gamma_t \left(\frac{1}{M} \sum_{m=1}^{M} \max_{\Delta x, \Delta \alpha} |\langle \mathbf{I}_m, B_{x_i+\Delta x, s_i, \alpha_i+\Delta\alpha} \rangle| - \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x_i,s_i,\alpha_i} \rangle| \right). \tag{20}$$

The learned $p(\mathbf{I}; \mathbf{B}, \lambda)$ models the appearance of the undeformed template. There is no local max pooling on the synthesized images, which have not undergone shape deformations or warping. The local max pooling is only applied to the observed images to filter out the shape deformations in the observed images. Thus there is an explicit separation between appearance and shape variations.

Again the synthesized images can be sampled by the HMC algorithm. For HMC computation, the energy function is

$$U(\mathbf{I}) = -\sum_{i=1}^{n} \lambda_i |\langle \mathbf{I}, B_{x_i,s_i,\alpha_i} \rangle| + \frac{1}{2}|\mathbf{I}|^2, \tag{21}$$

and the gradient of this energy function is

$$\frac{\partial U}{\partial \mathbf{I}} = -\sum_{i=1}^{n} \lambda_i \text{sign}(\langle \mathbf{I}, B_{x_i,s_i,\alpha_i} \rangle) B_{x_i,s_i,\alpha_i} + \mathbf{I}, \tag{22}$$

so HMC is like a generative process based on linear superpositions of $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$. With the separation between appearance and shape, the learned model for appearance may not be very multi-modal, therefore the HMC sampling can be quite fast.

Algorithm 3 Stage 2: Parameter estimation in sparse model

Input:
(1) training images $\{\mathbf{I}_m, m = 1, ..., M\}$,
(2) selected basis functions $\mathbf{B} = \{B_{x_i,s_i,\alpha_i}, i = 1, ..., n\}$
(3) inferred perturbations $\{\Delta x_{m,i}, \Delta\alpha_{m,i}, m = 1, ..., M, i = 1, ..., n\}$ by local max pooling.

Output:
$\lambda = \{\lambda_i, i = 1, ..., n\}$ and $\log Z(\lambda)$

1: Initialize $\lambda_i^{(0)} \leftarrow 0, i = 1, ..., n$.
2: Calculate observed statistics:
$H_i^{obs} \leftarrow \frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}} \rangle|$, for $i = 1, ..., n$.
3: Initialize synthesized images $\tilde{\mathbf{I}}_m$ as Gaussian white noises images
4: Initialize $\log Z(\lambda^{(0)}) \leftarrow 0$
5: Let $t \leftarrow 0$
6: **repeat**
7: Generate $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ from $p(\mathbf{I}; \mathbf{B}, \lambda^{(t)})$ by HMC
8: Calculate synthesized statistics:
$H_i^{syn} \leftarrow \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x_i,s_i,\alpha_i} \rangle|$, for $i = 1, ..., n$.
9: Update $\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \gamma_t (H_i^{obs} - H_i^{syn}), i = 1, ..., n$.
10: Compute Z ratio $\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$ by Eq. (13)
11: Update $\log Z(\lambda^{(t+1)}) \leftarrow \log Z(\lambda^{(t)}) + \log \frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$
12: Let $t \leftarrow t + 1$
13: **until** $\sum_i |H_i^{obs} - H_i^{syn}| \leq \epsilon$

The algorithm is presented in Algorithm 3. After we learn λ and compute $Z(\lambda)$ as in (13), we can use the learned model as a deformable template to be matched to the testing image, where the template matching score can be computed according to (19).

Figure 5 illustrates the basic idea of training the sparse FRAME model. The training images are scaled to 100 × 100. The number of selected basis functions (Gabor and large DoG wavelets), n, is set at 300. In principle it can be automatically determined by criteria like BIC. In the first stage, by using the deformable shared matching pursuit algorithm (see Algorithm 2) on the training images, we select n wavelets $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$, which are displayed in the first row, where each B_{x_i,s_i,α_i} is symbolized by a bar. The first four plots in the first row display the selected B_{x_i,s_i,α_i} at 4 different scales s_i, from the largest to the smallest. The last plot in the first row is a superposition of the 4 scales, with smaller scales appearing darker. The next four rows of the figure display four training images $\mathbf{I}_m$, the symbolic sketches of the deformed templates, $\mathbf{B}_m = (B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}}, i = 1, ..., n)$, the reconstructed images obtained by the linear superpositions of the perturbed basis functions, $\sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}}$,

and the residual image ϵ_m. At the second stage, we fit the sparse FRAME model with the n selected wavelets (see Algorithm 3). The synthesized images $\tilde{\mathbf{I}}_m$ generated from the learned model $p(\mathbf{I}; \mathbf{B}, \lambda)$ are projected onto the subspace spanned by $\mathbf{B}$. The last row displays projections of four synthesized images. These synthesized images show the appearances before shape deformations. Figure 6 shows another example.

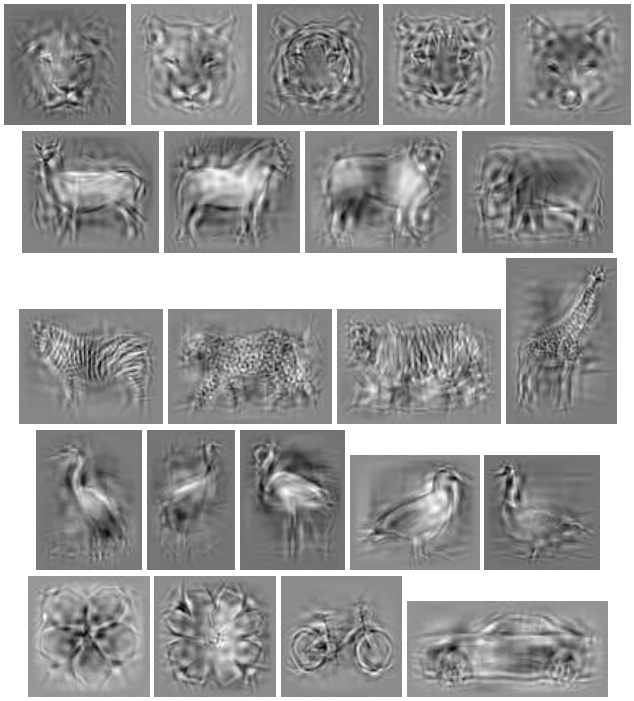

Fig. 7: Synthesis by sparse FRAME. Images generated by the sparse FRAME models learned from different categories of objects. Typical sizes of the images are 80×80. Typical number of selected wavelets is 300. The training images are the same as or similar to those for training the dense FRAME models in Fig. 4.

Experiment 2: Synthesis by Sparse FRAME. Figure 7 displays some images generated by the sparse models learned from roughly aligned images. The experiment setting is the same as that in Figure 5 except that the image sizes are typically 80×80, and the allowed displacement of Gabor wavelet is up to 2 pixels. The number of wavelets is 300. We run $\tilde{M} = 36$ parallel chains in the learning algorithm. Even though the number of wavelets are greatly reduced compared to the dense model, the sparse model can still generate realistic object patterns, including highly textured patterns. Because of the relatively small number of parameters, it is unlikely that the model memorizes the training images.

As to the actual running time, for the cat example, with 12 training images, the shared matching pursuit in stage 1 takes 95 seconds. For the algorithm of learning λ in stage 2, each iteration takes 2.8 second. The total running time is 6.5 minutes.

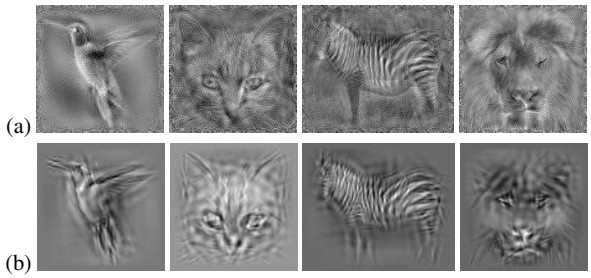

Fig. 8: Comparison of synthesized images generated by (a) dense FRAME and (b) sparse FRAME, where the number of selected wavelets is 300. Image sizes are about 100×100.

For a comparison of different models and learning methods, Figure 8 displays synthesized images generated by the dense FRAME and the sparse FRAME respectively.

4 Detection

After learning the sparse FRAME model $p(\mathbf{I}; \mathbf{B}, \lambda)$, where $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ and $\lambda = (\lambda_i, i = 1, ..., n)$, from the roughly aligned training images, we can use the learned model to detect the object in a testing image by deformable template matching.

Let $\mathbf{I}$ be a testing image defined on the domain $\mathcal{D}$. We can scan the template over $\mathcal{D}$, and at each location $X \in \mathcal{D}$, we match the template to the image patch of $\mathbf{I}$ within the bounding box centered at X by computing the log-likelihood or the template matching score based on (19),

$$L(\mathbf{I} \mid \mathbf{B}_X, \lambda) = \sum_{i=1}^{n} \lambda_i \max_{\Delta x, \Delta \alpha} |\langle \mathbf{I}, B_{X+x_i+\Delta x, s_i, \alpha_i+\Delta\alpha} \rangle| - \log Z(\lambda), \quad (23)$$

where we use $\mathbf{B}_X = (B_{X+x_i+\Delta x, s_i, \alpha_i+\Delta\alpha}, i = 1, ..., n)$ to denote the spatially translated and deformed version of the template $\mathbf{B}$. The perturbations of the basis functions are inferred by local max pooling as above. We then choose the location X that achieves the maximum template matching score as the center of the detected object. In practice, the template can be partially outside the image domain $\mathcal{D}$ when we scan the template near the boundary of $\mathcal{D}$. In this case, we only need to set the filter responses outside $\mathcal{D}$ to be zero. Also, to deal with the scaling issue, we can apply the above algorithm at multiple resolutions of the testing image, and then choose the resolution that achieves the maximum template matching score as the optimal resolution.

In addition to spatial translation in scanning, we can also allow geometric transformations such as rotation and left-right flipping of the template. The geometrically transformed

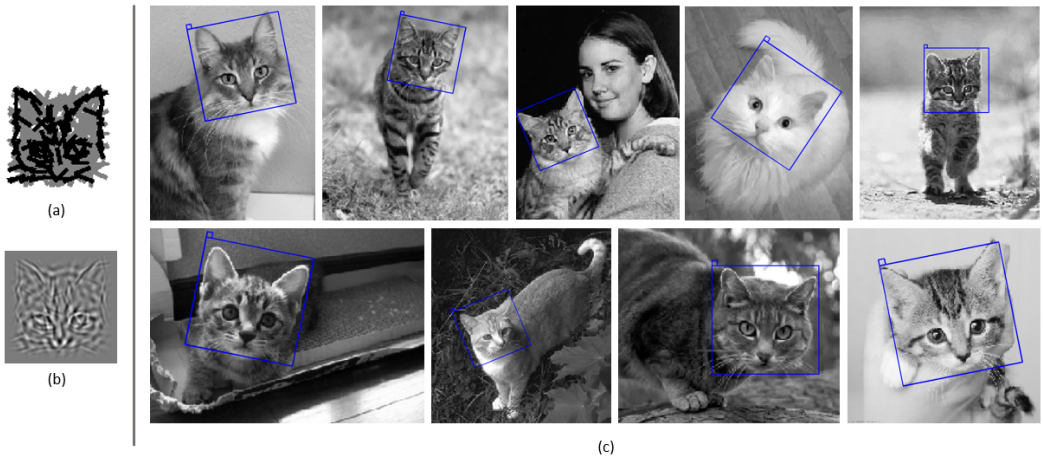

Fig. 10: Detection. (a) Symbolic sketch template representing 250 selected wavelets. (b) A synthesized image generated by the learned model. We do not include the large DoG filters in the model, so the synthesized image lacks regional contrast. (c) Testing images with bounding boxes locating the detected objects.

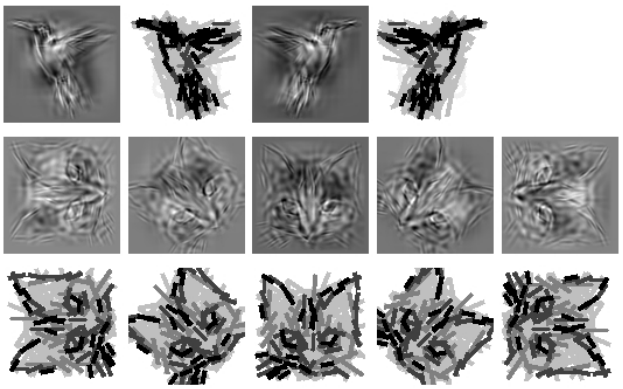

Fig. 9: Geometric transformation. Flipping: the first row shows an example of left/right flipping transformation, where the first two images are the synthesized image and the symbolic sketch template of the learned model, while the next two images correspond to the flipped model derived from the learned model. Rotation: the other example illustrated in the next two rows displays the rotated models at four different orientations (-90, -45, 45, and 90 degrees) by showing their synthesized images and symbolic sketches. The middle column is from the original learned model.

versions of the learned model can be obtained by directly applying the operations of dilation, rotation, flipping, or even changing the aspect ratio to $\mathbf{B} = \{B_{x_i,s_i,\alpha_i}, i = 1, ..., n\}$ without changing the values of λ. This amounts to simple affine transformations of $(x_i, s_i, \alpha_i, i = 1, ..., n)$. Figure 9 shows two examples of geometric transformations of the sparse FRAME model: flipping and rotation. It displays synthesized images generated by the transformed models, which are derived from the learned model, as well as the corresponding symbolic sketches representing the selected wavelets. For better performance in detection, we can first generate a collection of models at different orientations and aspect ratios from the learned model. After that, we use these transformed models to detect the object. We choose the combination of the transformed template and image resolution that gives the best match in terms of the template matching score, and infer the hidden location, orientation, and scale of the detected object in the testing image.

Experiment 3: Detection by sparse FRAME. Figure 10 shows examples of detection. We learn the model from eight roughly aligned training images, with $\tilde{M} = 36$. The template size is 100×100. The two images displayed in Figure 10(a) and 10(b) are symbolic sketches showing 250 wavelets selected by deformable shared matching pursuit algorithm and a synthesized image generated by the learned model. We transform the learned model into a collection of models at 9 different orientations, and then run the detection algorithm over 17 resolutions of the testing images using these transformed templates. Figure 10(c) displays the detection results by drawing bounding boxes on the detected objects.

This detection algorithm can be combined with the two-stage learning algorithm to learn from training images that are not well aligned by alternating the following two steps. (1) Re-learning the model from the currently aligned training images by the two-stage algorithm. (2) Re-aligning the training images by the detection algorithm.

5 Clustering

Model-based clustering can be accomplished by the EM-type algorithm [9] that fits mixtures of sparse FRAME mod-

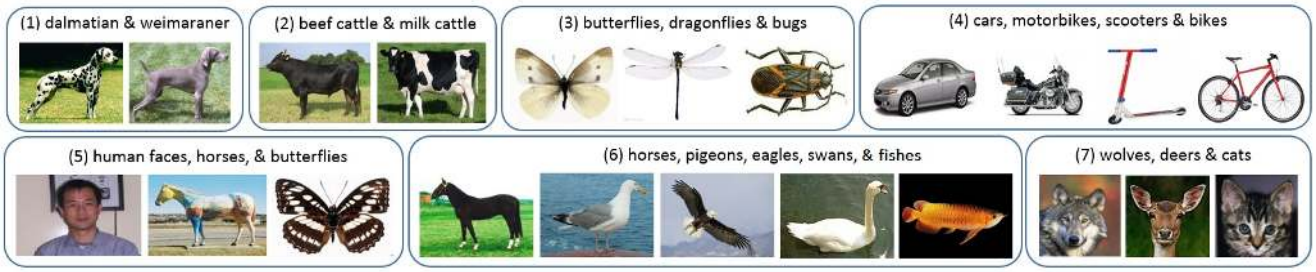

Fig. 12: The clustering dataset. One example image is shown for each of the 22 clusters distributed in 7 clustering tasks.

Table 1: Comparison of conditional purity (the first two rows) and conditional entropy (the last two rows) between sparse FRAME and k-means for clustering.

	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	Exp 7
k-means (purity)	0.623±0.016	0.870±0.043	0.933±0.141	0.825±0.121	0.911±0.086	0.888±0.091	0.687±0.110
FRAME (purity)	0.943±0.063	0.990±0.016	0.938±0.131	0.895±0.132	1.000±0.000	0.879±0.141	0.741±0.111
k-means (entropy)	0.652±0.009	0.376±0.086	0.092±0.195	0.243±0.167	0.226±0.084	0.199±0.126	0.639±0.161
FRAME (entropy)	0.145±0.157	0.037±0.060	0.090±0.191	0.155±0.189	0.000±0.000	0.179±0.208	0.497±0.192

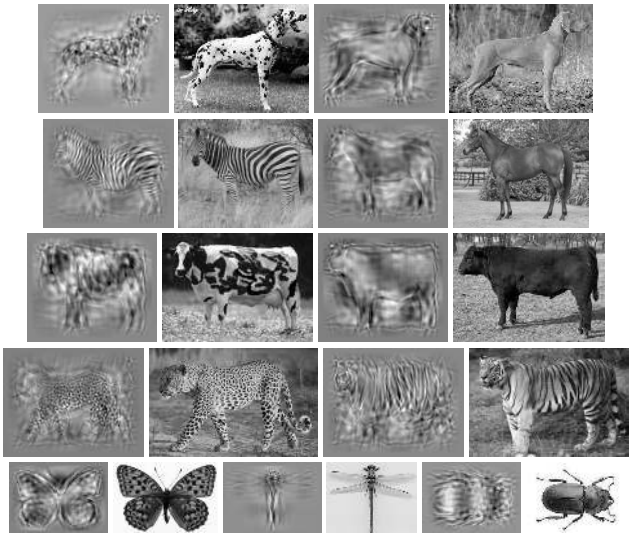

Fig. 11: Clustering. Each row illustrates one clustering experiment by displaying a synthesized image and a training example for each cluster. The number of images within each cluster is around 15 to 20. Typical template sizes are 100×100. Typical number of wavelets for each template is 300.

els. Suppose we have M images from K clusters. For each image $\mathbf{I}_m$, we define $(z_m^{(k)}, k = 1, ..., K)$ as a hidden indicator vector, where $z_m^{(k)} = 1$ if $\mathbf{I}_m$ comes from cluster k, otherwise $z_m^{(k)} = 0$. The EM-like clustering algorithm is a greedy scheme that infers $\{z_m^{(k)}\}$ and $\{\mathbf{B}^{(k)}, \lambda^{(k)}, k = 1, ..., K\}$ by maximizing the overall log-likelihood

$$\sum_{m=1}^{M} \sum_{k=1}^{K} z_m^{(k)} L(\mathbf{I}_m | \mathbf{B}^{(k)}, \lambda^{(k)}), \tag{24}$$

where $\mathbf{B}^{(k)}$ are the basis functions selected for cluster k, $\lambda^{(k)}$ are the corresponding parameters, and $L(\mathbf{I}_m | \mathbf{B}^{(k)}, \lambda^{(k)})$ is the log-likelihood or template matching score defined by (19).

The algorithm is initialized by randomly generating $\{z_m^{(k)}\}$, and then iterates the following two steps:

Re-learning: Given $\{(z_m^{(k)}, k = 1, ..., K), m = 1, ..., M\}$, learn the sparse FRAME model $p(\mathbf{I}; \mathbf{B}^{(k)}, \lambda^{(k)})$ from images classified into the k-th cluster: $\{\mathbf{I}_m, z_m^k = 1\}$, for each $k = 1, ..., K$.

Classification: Given the learned models of the K clusters: $\{p(\mathbf{I}; \mathbf{B}^{(k)}, \lambda^{(k)}), k = 1, ..., K\}$, assign each image $\mathbf{I}_m$ to a cluster k_* that maximizes the template matching score $L(\mathbf{I}_m | \mathbf{B}^{(k)}, \lambda^{(k)})$ over all $k = 1, ..., K$. Set $z_m^{(k_*)} = 1$, and set $z_m^{(k)} = 0$ for $k \neq k_*$.

In the above algorithm, the classification step corresponds to the E-step of the EM algorithm, except that we adopt hard classification instead of computing the expectation of z_m for each image $\mathbf{I}_m$. The re-learning step corresponds to the M-step of the EM algorithm. The algorithm usually converges within a few iterations.

Experiment 4: Model-based clustering. Figure 11 illustrates 5 experiments. The EM-type algorithm usually converges within 3-5 iterations, at which point all the images are correctly separated into their respective clusters. For each cluster, we generate $\tilde{M} = 144$ parallel chains in learning because we need to compute $Z(\lambda)$ accurately for each model,

as multiple models compete to explain the images. The same $\tilde{M} = 144$ is used for the experiments in the remaining part of the paper.

Experiment 5: Numerical evaluation on clustering. To evaluate the clustering accuracies, we use two measures: conditional purity and conditional entropy [56]. For a random training image, let x be its true category label and y be the inferred category label. The conditional purity is defined as $\sum_y p(y) \max_x p(x|y)$ (the larger the better), and the conditional entropy is $\sum_y p(y) \sum_x p(x|y) \log(1/p(x|y))$ (the smaller the better), where both $p(y)$ and $p(x|y)$ can be estimated from the training data. We also introduce a new dataset for clustering. Figure 12 provides an overview of the dataset. It contains 7 clustering tasks. The numbers of clusters vary from 2 to 5 and are assumed known in these tasks. The number of images in each cluster is typically 15 except in one experiment. We compare the performance of the sparse FRAME with that of the K-means method based on HoG [8] features by performing experiments on these 7 clustering tasks. Table 1 displays the clustering accuracies and standard errors based on 10 repetitions of each experiment.

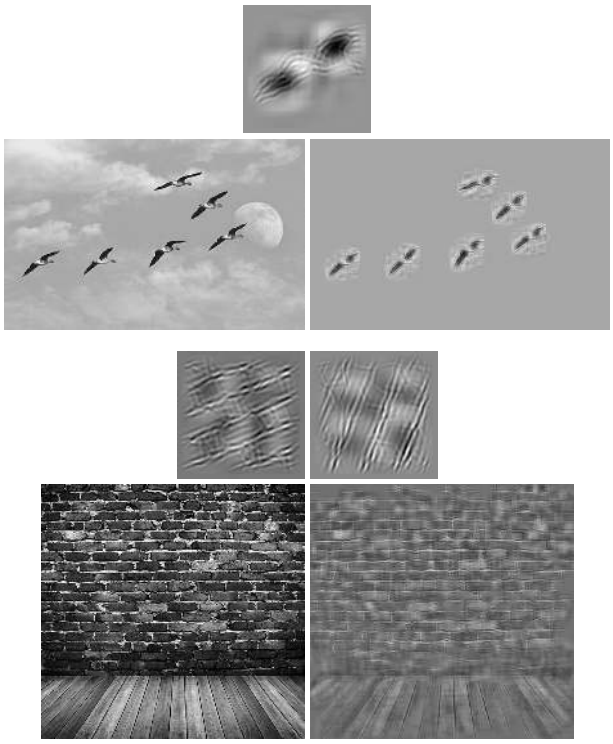

Fig. 13: Unsupervised learning. In each experiment, a codebook of sparse FRAME templates (of size 100×100 pixels) is learned from the training image. The images on the top row are generated from the learned templates. The image on the left of the second row is the observed training image. The image on the right of the second row is reconstructed using spatially translated, rotated and deformed versions of the learned templates.

6 Unsupervised learning from non-aligned images

In the previous section, we consider learning a single sparse FRAME model or template from roughly aligned images. The two-stage learning algorithm can serve as a basis for learning a codebook of sparse FRAME templates from non-aligned images without any annotation and labeling, so that the training images can be represented by spatially translated, rotated, scaled and deformed versions of templates selected from the learned codebook. Here we follow the learning scheme in our previous work on compositional sparse coding [29].

6.1 Learning a codebook of sparse FRAME models

Figure 13 shows two experiments. In the first experiment, a single template is learned. In the second experiment, a codebook of two templates (brick and floor tile patterns) are learned. In each experiment, the images on the top row are generated from the learned models. The image on the left of the second row is the observed image, and the image on the right of the second row is reconstructed by the learned templates.

Single template with spatial translation. To fix the notation, we shall assume temporarily that the templates are only allowed spatial translations in encoding the training images. We start from generalizing the representation (17) by assuming that the template may appear at location X_m in image $\mathbf{I}_m$, then we can write the representation as

$$\mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{X_m + x_i + \Delta x_{m,i}, s_i, \alpha_i + \Delta\alpha_{m,i}} + \epsilon_m \tag{25}$$

$$= C_m \mathbf{B}_{X_m} + \epsilon_m, \tag{26}$$

where $\mathbf{B}_{X_m} = (B_{X_m + x_i + \Delta x_{m,i}, s_i, \alpha_i + \Delta\alpha_{m,i}}, i = 1, ..., n)$ is the deformed template spatially translated to X_m, $C_m = (c_{m,i}, i = 1, ..., n)$, and by definition

$$C_m \mathbf{B}_{X_m} = \sum_{i=1}^{n} c_{m,i} B_{X_m + x_i + \Delta x_{m,i}, s_i, \alpha_i + \Delta\alpha_{m,i}}. \tag{27}$$

$\mathbf{B}_{X_m}$ 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, the log-likelihood is

$$L(\mathbf{I}_m \mid \mathbf{B}_{X_m}) = \sum_{i=1}^{n} \lambda_i \max_{\Delta x, \Delta\alpha} |\langle \mathbf{I}_m, B_{X_m + x_i + \Delta x, s_i, \alpha_i + \Delta\alpha} \rangle| - \log Z(\lambda), \tag{28}$$

which is a slight generalization of (19) and which is the log-likelihood score (23) used for object detection. For notational simplicity, we drop λ in $L(\mathbf{I}_m \mid \mathbf{B}_{X_m})$. We always assume that λ is estimated by MLE.

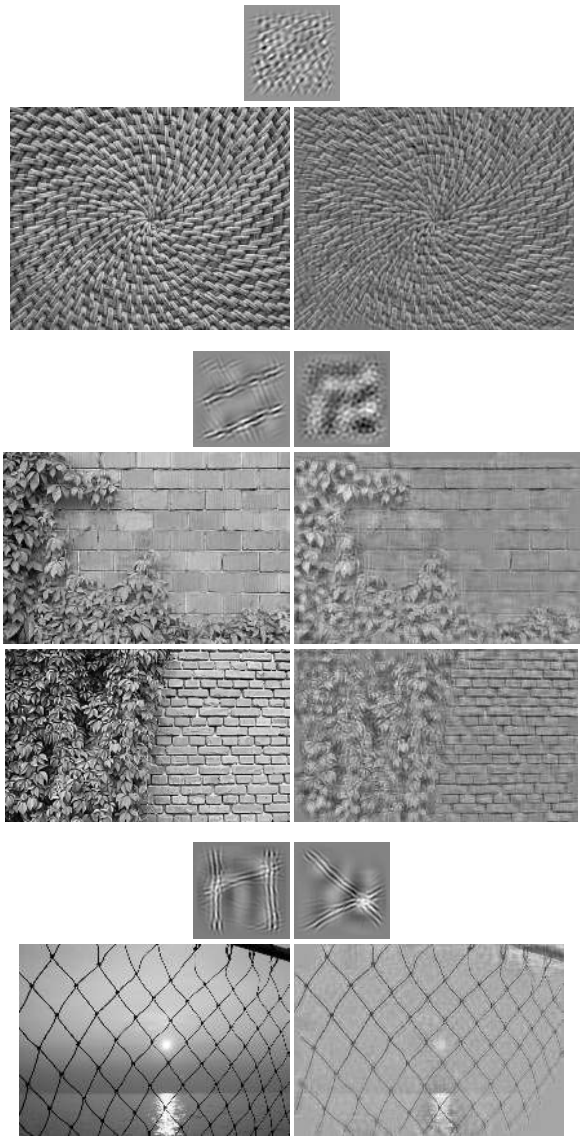

Fig. 14: Codebook learning. See caption of Fig. 13. In the second experiment (brick walls and ivy leaves), the image on the left of the third row is the testing image. The image on the right of the third row is reconstructed by the templates learned from the training image, which is on the left of the second row.

A codebook of templates and objective function. With the above notation such as that in (26), now suppose we have a codebook of T templates, and let us denote them by $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$. Then we can represent the image $\mathbf{I}_m$ by K_m templates that are spatially translated and deformed versions of these T templates in the codebook:

$$\mathbf{I}_m = \sum_{k=1}^{K_m} C_{m,k} \mathbf{B}^{(t_{m,k})}_{X_{m,k}} + \epsilon_m, \tag{29}$$

where each $\mathbf{B}^{(t_{m,k})}_{X_{m,k}}$ is obtained by translating the template of type $t_{m,k}$, i.e., $\mathbf{B}^{(t_{m,k})}$, to location $X_{m,k}$, and deforming it to match $\mathbf{I}_m$ by local max pooling in (28). For now, let us assume that the K_m templates do not overlap with each other, i.e., $\mathbf{B}^{(t_{m,k})}$ span orthogonal subspaces for $k = 1, ..., K_m$, such as in the first example of Figure 13. Then the dimensions that they explain are independent of each other, and the log-likelihood score is

$$L(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k}}, k = 1, ..., K_m) = \sum_{k=1}^{K_m} L(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k}}) \tag{30}$$

Our goal is to learn the codebook of T templates from the training images $\{\mathbf{I}_m\}$, while inferring the representation of each $\mathbf{I}_m$, i.e., $\{(t_{m,k}, X_{m,k}), k = 1, ..., K_m\}$, by maximizing the objective function which is defined as the sum of the log-likelihood (30) over all the training images $\{\mathbf{I}_m\}$,

$$\sum_{m=1}^{M} \left[\sum_{k=1}^{K_m} L(\mathbf{I}_m \mid \mathbf{B}^{(t_{m,k})}_{X_{m,k}}, k = 1, ..., K_m) \right], \tag{31}$$

subject to the constraint that for each $\mathbf{I}_m$, the encoding templates $\{\mathbf{B}^{(t_{m,k})}_{X_{m,k}}, k = 1, ..., K_m\}$ do not overlap.

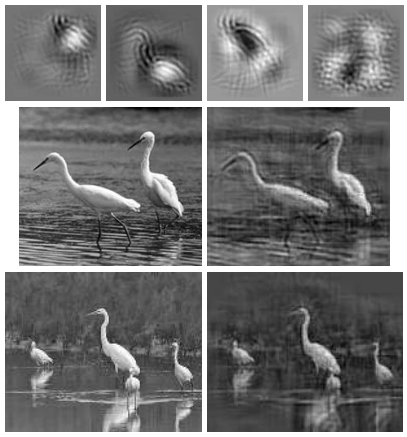

Fig. 15: Codebook learning. A codebook of 4 models (each has 250 wavelets) is learned from 20 images. The first row displays the synthesized images (100×100) from the 4 models. The second and third rows display two training images (left) and their reconstructions (right) by the 4 models.

Codebook learning algorithm. To initialize the unsupervised learning algorithm, we first learn the codebook of templates from randomly cropped image patches. Specifically, for each $\mathbf{B}^{(t)}$, we randomly cropped some image patches from training images, and then we learn $\mathbf{B}^{(t)}$ and the associated parameters $\lambda^{(t)}$ from these image patches using the two-stage algorithm described in the previous sections. Then we iterate the following two steps that seek to maximize the objective function (31):

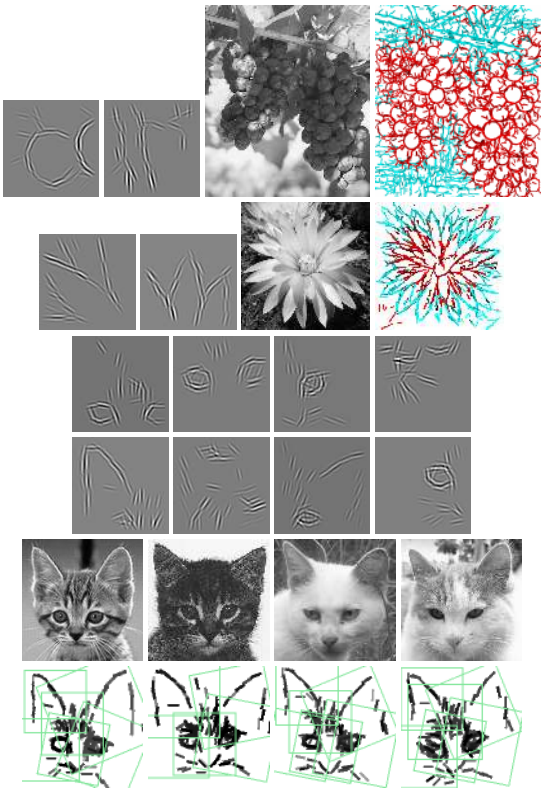

Fig. 16: Codebook learning. In each of the 3 experiments, synthesized images (100×100) from the models of the learned codebook are displayed together with the training images and their sketches by the learned models, where each Gabor wavelet is illustrated by a bar, and the templates appear in different colors (red and green) or with their bounding boxes (in green). Grape experiment: each model has 37 wavelets, learned from 1 image. Lotus experiment: each model has 30 wavelets, learned from 7 images. Cat experiment: each model has 40 wavelets, learned from 20 images.

(1) *Image encoding by template matching pursuit*. This step assumes that the codebook $\{\mathbf{B}^{(t)}\}$ is given, and it seeks to maximize (31) over the encoding of each image $\mathbf{I}_m$ by $\{(t_{m,k}, X_{m,k}), k = 1, ..., K_m\}$. Specifically, for each template in the codebook, we scan it over each image $\mathbf{I}_m$ and compute the log-likelihood score, i.e., compute $\mathbf{R}_m^{(t)}(X) = L(\mathbf{I}_m \mid \mathbf{B}_X^{(t)})$ for all t and X. Starting from $k = 1$, we sequentially select $(X_{m,k}, t_{m,k}) = \arg\max_{X,t} \mathbf{R}_m^{(t)}(X)$ subject to the constraint that $\mathbf{B}_{X_{m,k}}^{(t_{m,k})}$ does not overlap with previously selected templates and that the log-likelihood score of the selected template is above a threshold such as 0.

(2) *Template re-learning*. This step assumes that the encoding of each image $\mathbf{I}_m$, i.e., $\{(t_{m,k}, X_{m,k})\}$, is given, and it seeks to maximize (31) by re-learn the codebook of templates $\{(\mathbf{B}^{(t)}, \lambda^{(t)}), t = 1, ..., T\}$. Specifically, for each template t in the codebook, we re-learn $(\mathbf{B}^{(t)}, \lambda^{(t)})$ from the image patches currently encoded by this template using the two-stage learning algorithm.

The above algorithm is a greedy algorithm for maximizing the objective function (31). In fact, it can be considered a combination of detection and clustering studied in the previous sections. Even though the initial templates are random and meaningless, meaningful templates can usually be learned after a small number of iterations. These templates seek to explain different patterns in the observed images. According to our experience, meaningful templates can usually be learned regardless of the starting point of the algorithm.

In practical implementation of the above learning algorithm, we allow the templates to vary their rotations and scales in addition to spatial translation. We also allow the templates to have limited overlap, that is, after each template is selected, it only inhibits other templates within a limited distance from its center. Our experiences show that prohibiting overlap between the selected templates can result in parts of images left unexplained. Allowing limited overlap can avoid this problem.

In the template matching pursuit process, when a template is selected, it explains away part of the residual image by least squares projection. So after the template matching pursuit process, the observed images are reconstructed according to (29).

The number of templates in the codebook as well as the numbers of basis functions in the templates can be selected by BIC-like criteria, as suggested by [29]. In this paper, we hand picked these parameters.

Experiment 6: Unsupervised learning of codebooks. We can learn a codebook of sparse FRAME models from non-aligned images without annotation. Figure 14 illustrates 3 experiments of codebook learning. In each experiment, the images on the top row are synthetic images generated by the learned models. The input image is shown on the left of the second row. The image on the right of the second row is the reconstructed image using the learned templates. In the second experiment of brick walls and ivy leaves, the templates are learned from the training image in the second row, and they can be used to reconstruct the testing image in the third row. Figure 15 displays another example of a codebook learned from multiple images. Figure 16 displays results from another set of experiments, where for the sake of efficiency, we select $n = 40$ Gabor wavelets of a single scale, so the synthesized images mainly capture the edge patterns. Each experiment displays 100×100 images synthesized by the models in the learned codebook, together with the training image and the sketch of the image by the learned models (in different colors in the first two experiments or with green bounding boxes in the last experiment). There is one training image in the first experiment, while there are multiple training images in the other two experi-

ments. As to the running time, for the lotus example, each encoding and re-learning iteration takes about 2.6 minutes. We run 15 iterations.

6.2 Using learned codebooks for object classification

The learned codebook of sparse FRAME models can serve as "words" in the "bag-of-word" method for object classification. Suppose we have a codebook of T models $\{\mathbf{B}^{(t)}, t = 1, ..., T\}$ learned from training images. For each image $\mathbf{I}_m$, we denote $\mathbf{R}^{(t)}_m(X, S, A) = L(\mathbf{I}_m \mid \mathbf{B}^{(t)}_{X,S,A})$ as the log-likelihood of $\mathbf{B}^{(t)}$ at location X, scale S, and orientation A. Both S and A are assumed to take values within a finite and properly discretized range. Let

$$r^{(t)}_m(A) = \max(\max_{X,S} R^{(t)}_m(X, S, A), 0) \tag{32}$$

be the maximum score at orientation A. Then each image $\mathbf{I}_m$ can be represented by a $T \times N_A$-dimensional feature vector $(r^{(t)}_m(A), t = 1, ..., T, \forall A)$, where N_A is the number of possible values A can take. After extracting features, we can use any discriminative method to train classifiers (e.g. linear logistic regression or SVM [57]) on such feature vectors for object classification. Spatial pyramid matching (SPM) [31] can also be utilized to further boost the classification performance.

Experiment 7: Binary classification. We evaluate the above "bag-of-word" representation extracted by a codebook of sparse FRAME templates on a binary classification task. We test it on a collection of 16 categories from Caltech-101 [15], all 5 categories from ETHZ Shape [17] and all 3 categories from Graz-02 [37] datasets. The task is to separate each category from a negative background category. We resize all images to 150×150 pixels without changing their aspect ratios and convert them to grey level images. We randomly choose 30 positive and 30 negative images respectively as training data, and keep the rest as testing data. For Caltech-101 and Graz-02, negative images are chosen from the background category, while for ETHZ, negative examples are chosen from images other than the target category. For each category, we learn a codebook of $T = 10$ sparse FRAME templates. Each template is of the size 100×100 and has $n = 40$ wavelets. We set scale $S \in \{0.8, 1, 1.2\}$ and orientation $A \in \{\pm 1, 0\} \times \pi/16$. Binary classification is done with linear logistic regression with regularization by ℓ_2 norm [14]. We compare our results with those obtained by SIFT [35] features and SVM classifier, where SIFT features are quantized into "words" by K-means clustering (K = 50, 100, 500) and fed into linear or kernel SVM. The best result among these six combinations (3 numbers of words $\times$ two types of SVM) is then reported. Table 2 shows the comparison results of the binary classification experiments.

Table 3: Classification accuracies (%) on the animal faces dataset.

HoG+SVM	HIT	Mixture of HIT	Part-based LSVM	Our method
70.8	71.6	75.6	77.6	**79.4**

All experiments are repeated five times with different randomly selected training and testing images, and the average accuracies and the 95% confident intervals are reported. It can be seen that our method generally outperforms the SIFT + SVM method, despite the fact that we use much smaller codebooks (10 "words" versus 50, 100, 500 "words").

Experiment 8: Multi-class classification. Our second set of experiments is on the LHI-Animal-Faces dataset [51], which consists of around 2200 images for 20 categories of animal or human faces. We randomly select half of the images per class for training and the rest for testing. We learn a codebook of 10 sparse FRAME models for each category in an unsupervised way. We then combine the codebooks of all the categories (in total $20 \times 10 = 200$ codewords). The maps of the template matching scores from the models in the combined codebook are computed for each image, and they are then fed into SPM, which equally divides an image into 1, 4, 16 areas, and concatenates the maximum scores at different image areas into a feature vector. We use multi-class SVM to train image classifiers based on the feature vectors, and then evaluate the classification accuracies of these classifiers on the testing data using the one-versus-all rule. Our classification rate is 79.4%. For comparison, Table 3 lists four published results [51] on this dataset obtained by other methods: (a) HoG feature trained with SVM, (b) Hybrid Image Template (HIT) [51], (c) multiple transformation invariant HITs (Mixture of HIT) [51], and (d) part-based HoG feature trained with latent SVM [16]. Our method outperforms the other methods in terms of classification accuracy on this dataset.

Experiment 9: Domain transfer. Classifiers learned from one domain (the source domain) may perform poorly on other domains (the target domains), because that training and testing data may not come from the same distribution. Learning domain-invariant feature representations can deal with this problem. In this experiment, we test our proposed representation for the task of domain transfer on four domain datasets, and compare with published results [49] [24] [23] [61] [30] [28] [50]. The four datasets are: Amazon, Webcam, DSLR and Caltech-256 [25]. Each dataset is regarded as a domain. For the experiment with single source training, 10 classes common to all 4 datasets are extracted: backpack, touring-bike, calculator, head-phones, computer-keyboard, laptop-101, computer-monitor, computer-mouse, coffee-mug, and video-projector. For the experiment with

Table 2: 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**	89.1±1.6	Sunflower	76.0±2.5	**89.6±3.7**
Laptop	73.5±5.3	**89.8±2.7**	Chair	62.5±5.0	**82.9±4.7**
Piano	84.5±4.2	**93.8±2.6**	Lamp	61.5±4.5	**86.6±4.3**
Ketch	82.2±0.8	**83.3±6.5**	Dragonfly	66.0±4.0	**89.9±5.7**
Motorbike	**93.9±1.2**	92.2±2.9	Umbrella	73.4±4.4	**90.0±0.7**
Guitar	70.0±2.4	**77.3±6.3**	Cellphone	68.7±5.1	**95.7±1.8**
Schooner	64.3±2.2	**87.7±2.8**	Face	91.8±2.3	**94.4±2.3**
Ibis	67.8±6.0	**85.3±2.7**	Starfish	73.1±6.7	**90.0±2.3**
ETHZ-Bottle	68.6±3.2	**77.5±5.6**	ETHZ-Cup	**66.0±3.3**	62.5±3.0
ETHZ-Swans	64.2±1.5	**74.0±7.5**	ETHZ-Giraffes	61.5±6.4	**73.3±4.8**
ETHZ-Apple	55.0±1.8	**65.8±6.1**	Graz02-Person	**70.4±1.2**	68.2±3.8
Graz02-Car	**64.0±6.7**	59.6±5.5	Graz02-Bike	68.5±2.8	**71.3±5.1**

Table 4: Results on the domain transfer experiment

(a) Classification accuracies (%) on single source four domains benchmark (C: caltech, A: amazon, D: DSLR, W: webcam)

Method	C→A	C→D	A→C	A→W	W→C	W→A	D→A	D→W
Metric [49]	33.7±0.8	35.0±1.1	27.3±0.7	36.0±1.0	21.7±0.5	32.3±0.8	30.3±0.8	55.6±0.7
SGF [24]	40.2±0.7	36.6±0.8	37.7±0.5	37.9±0.7	29.2±0.7	38.2±0.6	39.2±0.7	69.5±0.9
GFK [23]	46.1±0.6	55.0±0.9	39.6±0.4	56.9±1.0	32.8±0.7	46.2±0.7	46.2±0.6	**80.2±0.4**
FDDL [61]	39.3±2.9	55.0±2.8	24.3±2.2	50.4±3.5	22.9±2.6	41.1±2.6	36.7±2.5	65.9±4.9
MMDT [28]	49.4±0.8	56.5±0.9	36.4±0.8	64.6±1.2	32.2±0.8	47.7±0.9	46.9±1.0	74.1±0.8
SDDL [50]	49.5±2.6	**76.7±3.9**	27.4±2.4	**72.0±4.8**	29.7±1.9	49.4±2.1	48.9±3.8	72.6±2.1
Our method	**62.2±1.6**	52.2±4.0	**46.7±2.5**	53.2±4.9	**39.1±3.0**	**53.2±4.4**	**55.3±2.9**	72.4±3.1

(b) Classification accuracies (%) on multiple sources three domains benchmark

Source	Target	SGF [24]	RDALR [30]	FDDL [61]	Our method
DLSR, amazon	webcam	52±2.5	36.9±1.1	41.0±2.4	**52.2±1.4**
amazon, webcam	DSLR	39±1.1	31.2±1.3	38.4±3.4	**54.5±3.3**
webcam, DSLR	amazon	28±0.8	20.9±0.9	19.0±1.2	**32.1±1.6**

multiple sources training, all 31 classes in Amazon, Webcam and DSLR are used. We use the evaluation protocol in [23]. We randomly sample labeled data in the source domain as training examples, and unlabeled data in the target domain as testing examples. We learn a combined codebook (by learning a codebook of 3 templates with $n = 40$ wavelets for each category and combining them together), then use it to extract feature vectors and train classifiers by multi-class SVM using the same scheme as in Experiment 8. We evaluate the classification accuracies of these classifiers on the testing domain. For each pair of source and target domains, we report averaged accuracies on target domains as well as standard errors. Table 4 shows the comparisons of recognition accuracies on target domains for single source training and multiple source training, where the accuracies and standard errors are obtained from 10 repetitions. It can be seen that our method performs significantly better than previous methods on 8 out of 11 sub-tasks, and on-par with the best performing method on the other sub-tasks, even though we do not make use of any domain adaptation techniques. This suggests that the learned codebooks of models capture intrinsically meaningful patterns.

7 Conclusion

We propose that the sparse FRAME models form the layer of representational units above the layer of wavelets sparse coding. A sparse FRAME model makes use of wavelet sparse coding to generate image intensities, while accounting for the distribution of the coefficients of the selected wavelets as well as perturbations of their locations and orientations.

As a generative model, the sparse FRAME model has the following characteristics. (1) It can reconstruct the training images, and reconstruction is used for selecting the basis

functions. (2) It can synthesize new images, and synthesis is required for estimating parameters and calculating the normalizing constant. (3) It separates shape deformations and appearance variations. (4) It gives interpretable sketches. (5) Codebooks of models can be learned in an unsupervised manner. (6) It combines rich traditions of harmonic analysis and Markov random field models.

While we have shown that it is possible to learn codebooks of sparse FRAME models, much remains to be understood about learning large codebooks reliably from big training data sets.

Reproducibility

http://www.stat.ucla.edu/~jxie/sparseFRAME.html
The above webpage contains the full data sets and exact parameter settings, and matlab/C code for producing the experimental results presented in this paper.

Acknowledgements The work is supported by NSF DMS 1310391, NSF IIS 1423305, ONR MURI N00014-10-1-0933, DARPA MSEE FA8650-11-1-7149. We thank the three reviewers for their insightful comments and valuable suggestions that have helped us improve the presentation and the content of this paper. We are grateful to one reviewer for sharing the insights on the analysis prior models. Thanks also go to an editor of the special issue for helpful suggestions. We thank Adrian Barbu for discussions.

8 Appendix

8.1 Simulation by Hamiltonian Monte Carlo

To approximate $\mathrm{E}_{p(\mathbf{I};\lambda^{(t)})}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|]$ in equation (9), we need to draw a synthesized sample set $\{\tilde{\mathbf{I}}_m\}$ from $p(\mathbf{I};\lambda^{(t)})$ by HMC [10]. We can write $p(\mathbf{I};\lambda)$ as $p(\mathbf{I}) \propto \exp(-U(\mathbf{I}))$, where $\mathbf{I} \in R^{|\mathcal{D}|}$ and

$$U(\mathbf{I}) = -\sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}, B_{x,s,\alpha}\rangle| + \frac{1}{2}|\mathbf{I}|^2 \tag{33}$$

(assuming $\sigma^2 = 1$). In physics context, $\mathbf{I}$ can be regarded as a position vector and $U(\mathbf{I})$ the potential energy function. To allow Hamiltonian dynamics to operate, we need to introduce an auxiliary momentum vector $\boldsymbol{\phi} \in R^{|\mathcal{D}|}$ and the corresponding kinetic energy function $K(\boldsymbol{\phi}) = |\boldsymbol{\phi}|^2/2m$, where m represents the mass. After that, a fictitious physical system described by the canonical coordinates $(\mathbf{I}, \boldsymbol{\phi})$ is defined, and its total energy is $H(\mathbf{I}, \boldsymbol{\phi}) = U(\mathbf{I}) + K(\boldsymbol{\phi})$. Instead of sampling from $p(\mathbf{I})$ directly, HMC samples from the joint canonical distribution $p(\mathbf{I}, \boldsymbol{\phi}) \propto \exp(-H(\mathbf{I}, \boldsymbol{\phi}))$, under which $\mathbf{I} \sim p(\mathbf{I})$ marginally and $\boldsymbol{\phi}$ follows a Gaussian distribution and is independent of $\mathbf{I}$. Each time HMC draws a random sample from the marginal Gaussian distribution of $\boldsymbol{\phi}$, and then evolves according to the Hamiltonian dynamics that conserves the total energy.

In practical implementation, the leapfrog algorithm is used to discretize the continuous Hamiltonian dynamics as follows, with ϵ being the step-size:

$$\boldsymbol{\phi}^{(t+\epsilon/2)} = \boldsymbol{\phi}^{(t)} - (\epsilon/2)\frac{\partial U}{\partial \mathbf{I}}(\mathbf{I}^{(t)}), \tag{34}$$

$$\mathbf{I}^{(t+\epsilon)} = \mathbf{I}^{(t)} + \epsilon \frac{\boldsymbol{\phi}^{(t+\epsilon/2)}}{m}, \tag{35}$$

$$\boldsymbol{\phi}^{(t+\epsilon)} = \boldsymbol{\phi}^{(t+\epsilon/2)} - (\epsilon/2)\frac{\partial U}{\partial \mathbf{I}}(\mathbf{I}^{(t+\epsilon)}), \tag{36}$$

that is, a half-step update of $\boldsymbol{\phi}$ is performed first and then it is used to compute $\mathbf{I}^{(t+\epsilon)}$ and $\boldsymbol{\phi}^{(t+\epsilon)}$.

A key step in the leapfrog algorithm is the computation of the derivative of the potential energy function

$$\frac{\partial U}{\partial \mathbf{I}} = -\sum_{x,s,\alpha} \lambda_{x,s,\alpha} \mathrm{sign}(\langle \mathbf{I}, B_{x,s,\alpha}\rangle) B_{x,s,\alpha} + \mathbf{I}, \tag{37}$$

where the map of responses $r_{x,s,\alpha} = \langle \mathbf{I}, B_{x,s,\alpha}\rangle$ is computed by bottom-up convolution of the filter corresponding to (s,α) with $\mathbf{I}$ for each (s,α). Then the derivative is computed by top-down linear superposition of the basis functions: $-\sum_{x,s,\alpha} \lambda_{x,s,\alpha} \mathrm{sign}(r_{x,s,\alpha}) B_{x,s,\alpha} + \mathbf{I}$, which can again be computed by convolution. Both bottom-up and top-down convolutions can be carried out efficiently by GPUs.

The discretization of the leapfrog algorithm cannot keep $H(\mathbf{I}, \boldsymbol{\phi})$ exactly constant, so a Metropolis acceptance/rejection step is used to correct the discretization error. Starting with the current state, $(\mathbf{I}, \boldsymbol{\phi})$, the new state $(\mathbf{I}^\star, \boldsymbol{\phi}^\star)$, after L leapfrog steps, is accepted as the next state of the Markov chain with probability $\min[1, \exp(-H(\mathbf{I}^\star, \boldsymbol{\phi}^\star) + H(\mathbf{I}, \boldsymbol{\phi}))]$. If it is not accepted, the next state is the same as the current state.

In summary, a complete description of the HMC sampler for inhomogeneous FRAME is as follows:

(i) Generate the momentum vector $\boldsymbol{\phi}$ from its marginal distribution $p(\boldsymbol{\phi}) \propto \exp(-K(\boldsymbol{\phi}))$, which is the zero-mean Gaussian distribution with covariance matrix mI (I is the identity matrix).

(ii) Perform L leapfrog steps to reach the new state $(\mathbf{I}^\star, \boldsymbol{\phi}^\star)$.

(iii) Perform acceptance/rejection of the proposed state $(\mathbf{I}^\star, \boldsymbol{\phi}^\star)$.

L, ϵ, and m are parameters of the algorithm, which need to be tuned to obtain good performance.

8.2 Maximum entropy justification

The inhomogeneous FRAME model can be justified by the maximum entropy principle. Suppose the true distribution that generates the observed images $\{\mathbf{I}_m\}$ is $f(\mathbf{I})$. Let $\lambda^\star$ solve the population version of the maximum likelihood equation:

$$\mathrm{E}_{p(\mathbf{I};\lambda)}[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|] = \mathrm{E}_f[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|], \ \forall x, s, \alpha. \tag{38}$$

Let Ω be the set of all the distributions $p(\mathbf{I})$ such that

$$\mathrm{E}_p[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|] = \mathrm{E}_f[|\langle \mathbf{I}, B_{x,s,\alpha}\rangle|], \ \forall x, s, \alpha. \tag{39}$$

Then $f \in \Omega$. Let Λ be the set of all the distributions $\{p_\lambda, \forall \lambda\}$, where $p_\lambda(\mathbf{I}) = p(\mathbf{I}; \lambda)$. Then $q \in \Lambda$ since $q(\mathbf{I}) = p(\mathbf{I}; \lambda = 0)$. Thus $p_{\lambda^\star}$ is the intersection between Λ and Ω. In Figure 17, Λ and Ω are illustrated by blue and green curves respectively, where each point on the curves is a probability distribution. The two curves Λ and Ω are "orthogonal" in the sense that for any $p_\lambda \in \Lambda$ and for any $p \in \Omega$, it can be easily proved that the the Pythagorean property

$$\mathrm{KL}(p||p_\lambda) = \mathrm{KL}(p||p_{\lambda^\star}) + \mathrm{KL}(p_{\lambda^\star}||p_\lambda) \tag{40}$$

holds [44], where $\mathrm{KL}(p||q)$ is the Kullback-Leibler divergence from p to q. This Pythagorean property leads to the following dual properties of $p_{\lambda^\star}$:

(1) Maximum likelihood: Among all $p_\lambda \in \Lambda$, $p_{\lambda^\star}$ achieves the minimum of $\mathrm{KL}(f||p_\lambda)$.

(2) Maximum entropy or minimum divergence: Among all $p \in \Omega$, $p_{\lambda^\star}$ achieves the minimum of $\mathrm{KL}(p||q)$. Thus $p_{\lambda^\star}$ can be considered the minimal modification of the reference distribution q to match the statistical properties of the true distribution f.

The above justification is also true for the sparse FRAME model.

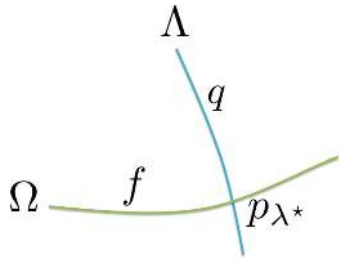

Fig. 17: Illustration of the maximum entropy principle. Each curve illustrates a set of probability distributions. Ω is the set of distributions that reproduce statistical properties of filter response of the true distribution f. Λ is the set of distributions of the model. The two curves are orthogonal to each other in the sense of the Pythagorean property of the Kullback-Leibler divergences. So $p_{\lambda^\star}$ can be considered the minimal modification of the reference distribution q to match the statistical properties of f.

For sparsification, in principle, we can select B_{x_i,s_i,α_i} sequentially using a procedure like projection pursuit [19] or filter pursuit [66]. Suppose we have selected k basis functions $(B_{x_i,s_i,\alpha_i}, i = 1, ..., k)$, and let p_k be the fitted model with the corresponding $\lambda = (\lambda_i, i = 1, ..., k)$ estimated by MLE. Suppose we are to select the next basis function $B_{x_{k+1},s_{k+1},\alpha_{k+1}}$. Let p_{k+1} be the fitted model. Then we want to minimize $\mathrm{KL}(f||p_{k+1}) = \mathrm{KL}(f||p_k) - \mathrm{KL}(p_{k+1}||p_k)$, that is, we want to maximize $\mathrm{KL}(p_{k+1}||p_k)$, which serves as the pursuit index. The problem with such a procedure is that each time we need to fit p_k which involves MCMC computation, and the pursuit index is also difficult to compute. So we choose to pursue a different approach by exploring the connection between sparse FRAME and the shared sparse coding.

8.3 Sparse FRAME and shared sparse coding

From sparse FRAME to shared sparse coding. Let us assume that the reference distribution $q(\mathbf{I})$ in the sparse FRAME model (15) is a Gaussian white noise model so that the pixel intensities follow $\mathrm{N}(0, \sigma^2)$ independently. For sparse FRAME, it is natural to assume that the number of selected basis functions n is much less than the number of pixels in $\mathbf{I}$, i.e., $n \ll |\mathcal{D}|$, where $\mathcal{D}$ is the image domain. For notational convenience, we can make $\mathbf{I}$ and $B_i = B_{x_i,s_i,\alpha_i}$, $i = 1, ..., n$ into $|\mathcal{D}|$-dimensional vectors, and let $\mathbf{B} = (B_1, ..., B_n)$ be the resulting $|\mathcal{D}| \times n$ matrix.

The connection between sparse FRAME and shared sparse coding is most evident if we temporarily assume that the selected basis functions $(B_i, i = 1, ..., n)$ are orthogonal (with unit ℓ_2 norm as assumed before). Extension to non-orthogonal $\mathbf{B}$ is straightforward but requires tedious notation (such as $(\mathbf{B}^T\mathbf{B})^{-1}$). For $\mathbf{B}$, we can construct $\bar{n} = |\mathcal{D}| - n$ basis vectors of unit norm $\bar{B}_1, ..., \bar{B}_{\bar{n}}$ that are orthogonal to each other and that are also orthogonal to $(B_i, i = 1, ..., n)$. Thus each image $\mathbf{I} = \sum_{i=1}^n r_i B_i + \sum_{i=1}^{\bar{n}} \bar{r}_i \bar{B}_i$, where $r_i = \langle \mathbf{I}, B_i \rangle$, and $\bar{r}_i = \langle \mathbf{I}, \bar{B}_i \rangle$. So we have the linear additive model $\mathbf{I} = \sum_{i=1}^n r_i B_i + \epsilon$, with $\epsilon = \sum_{i=1}^{\bar{n}} \bar{r}_i \bar{B}_i$ being the least squares residual image.

Under the Gaussian white noise $q(\mathbf{I})$, r_i and $\bar{r}_i$ are all independent $\mathrm{N}(0, \sigma^2)$ random variables because of the orthogonality of $(\mathbf{B}, \bar{\mathbf{B}})$. Let R be the column vector whose elements are r_i, and $\bar{R}$ be the column vector whose elements are $\bar{r}_i$. Then under the sparse FRAME model (15), only the distribution of R is modified during the change from $q(\mathbf{I})$ to $p(\mathbf{I}; \mathbf{B}, \lambda)$, which changes the distribution of R from Gaussian white noise $q(R)$ to

$$p(R; \lambda) = \frac{1}{Z(\lambda)} \exp\left(\sum_{i=1}^n \lambda_i |r_i| \right) q(R), \tag{41}$$

while the distribution of the residual coordinates $\bar{R}$ remains Gaussian white noise, and R and $\bar{R}$ remain independent. That is, $p(R, \bar{R}; \lambda) = p(R; \lambda) q(\bar{R})$.

Thus the sparse FRAME model implies a linear additive model $\mathbf{I} = \sum_{i=1}^n r_i B_i + \epsilon$, where $R \sim p(R; \lambda)$ and ϵ is a Gaussian white noise in the $\bar{n}$-dimensional residual space, and ϵ is independent of R. If we observe independent training images $\{\mathbf{I}_m, m = 1, ..., M\}$ from the model, then $\mathbf{I}_m = \sum_{i=1}^n r_{m,i} B_i + \epsilon_m$, i.e., $\{\mathbf{I}_m\}$ share a common set of basis functions $\mathbf{B} = (B_i, i = 1, ..., n)$ that provide sparse coding for multiple images simultaneously.

From shared sparse coding to sparse FRAME. Conversely, suppose we are given a shared sparse coding model of the form $\mathbf{I} = \sum_{i=1}^{n} c_i B_i + \epsilon = \mathbf{B}C + \epsilon$, where C is a column vector whose components are c_i. Assume $C \sim p(C)$ and $\epsilon \sim \mathrm{N}(0, I\sigma^2)$, where I is the $|\mathcal{D}|$-dimensional identity matrix, and ϵ and C are independent. Let $\delta = \mathbf{B}^T\epsilon$, each component of which $\delta_i = \langle \epsilon, B_i \rangle \sim \mathrm{N}(0, \sigma^2)$ independently. Then we can write $\mathbf{I} = \mathbf{B}R + \bar{\mathbf{B}}\bar{R}$, where $R = C + \delta$, and $\bar{\epsilon} = \bar{\mathbf{B}}\bar{R}$ is the projection of ϵ onto the space of $\bar{\mathbf{B}}$. Let $\tilde{p}(R)$ be the density of $R = C + \delta$, which is obtained by convolving $p(C)$ with Gaussian white noise density. Then $p(\mathbf{I}) = \tilde{p}(R)q(\bar{R}) = q(\mathbf{I})\tilde{p}(R)/q(R)$ since $q(\mathbf{I}) = q(R)q(\bar{R})$ under Gaussian white noise model ($d\mathbf{I} = dRd\bar{R}$ under orthogonality so there is no Jacobian term). If we choose to model $\tilde{p}(R)/q(R) = \exp\left(\sum_{i=1}^{n} \lambda_i |r_i|\right)/Z(\lambda)$, we arrive at the sparse FRAME model.

Selection of basis functions. For orthogonal $\mathbf{B}$, as shown above, the probability density $p(\mathbf{I}; \mathbf{B}, \lambda) = q(\bar{R})p(R; \lambda) = q(\bar{R})q(R) \exp\left(\sum_{i=1}^{n} \lambda_i |r_i|\right)/Z(\lambda)$. Given a set of training images $\{\mathbf{I}_m, m = 1, ..., M\}$, and for a candidate set of basis functions $\mathbf{B} = (B_i, i = 1, ..., n)$, we can estimate $\lambda = (\lambda_i, i = 1, ..., n)$ by MLE, giving us $\lambda^\star$, and the resulting log-likelihood is

$$\sum_{m=1}^{M} \log p(\mathbf{I}_m; \mathbf{B}, \lambda^\star)$$

$$= \sum_{m=1}^{M} \left[\log q(\bar{R}_m) + \log p(R_m; \lambda^\star)\right] \quad (42)$$

$$= -\frac{1}{2\sigma^2} \sum_{m=1}^{M} ||\mathbf{I}_m - \mathbf{B}R_m||^2 - \frac{M\bar{n}}{2} \log(2\pi\sigma^2) \quad (43)$$

$$+ \sum_{m=1}^{M} \log p(R_m; \lambda^\star). \quad (44)$$

Suppose we are to choose a $\mathbf{B}$ from a collection of candidates. Ideally we should maximize the sum of (43) and (44). We may interpret (43) to be the negative coding length of the residual image ϵ by the Gaussian white noise model, and interpret (44) to be the negative coding length of the coefficients R by the fitted model $p(R; \lambda^\star)$. If σ^2 is small, (43) can be more important, while the coding length of R for different $\mathbf{B}$ may not differ too much in comparison. So we choose to seek a $\mathbf{B}$ to maximize only (43) or equivalently minimize the overall reconstruction error $\sum_{m=1}^{M} ||\mathbf{I}_m - \mathbf{B}R_m||^2$. This reflects a two-stage strategy in modeling $\{\mathbf{I}_m\}$. First, we find a set of basis functions $\mathbf{B}$ to reconstruct $\{\mathbf{I}_m\}$ as accurately as possible. Then we fit a statistical model for the reconstruction coefficients.

Non-orthogonality. Even if $\mathbf{B}$ is not orthogonal, which is the case in our work, the connection between the sparse FRAME and shared sparse coding still holds. The responses $R = \mathbf{B}^T\mathbf{I}$, but the reconstruction coefficients become $C = (\mathbf{B}^T\mathbf{B})^{-1}R$. The projection of $\mathbf{I}$ onto the subspace spanned by $\mathbf{B}$ is $\mathbf{B}C$. We can continue to assume the implicit $\bar{\mathbf{B}} = (\bar{B}_i, i = 1, ..., \bar{n})$ to be orthonormal, and that they are orthogonal to the columns of $\mathbf{B}$. We can also continue to let $\bar{R} = \bar{\mathbf{B}}^T\mathbf{I}$. In this setting, R and $\bar{R}$ are still independent under the Gaussian white noise model $q(\mathbf{I})$ because $\mathbf{B}$ and $\bar{\mathbf{B}}$ are still orthogonal to each other. Under the sparse FRAME model (15), it is still the case that only the distribution of R is modified during the change from $q(\mathbf{I})$ to $p(\mathbf{I}; \mathbf{B}, \lambda)$, while the distribution of $\bar{R}$ remains white noise and is independent of R. The distribution of R implies a distribution of the reconstruction coefficients C because they are linked by a linear transformation. In fact, the distribution of C is:

$$p_C(C; \lambda) = \frac{1}{Z(\lambda)} \exp(\langle \lambda, |\mathbf{B}^T\mathbf{B}C| \rangle) q_C(C), \quad (45)$$

where $q_C(C)$ is the distribution of C under the reference distribution $q(\mathbf{I})$, and for a vector u, $|u|$ means the vector obtained by taking the absolute values of u component-wise. Now the distributions of R and C involve the Jacobian terms such that $dRd\bar{R} = |\det(\mathbf{B}^T\mathbf{B})|^{1/2} d\mathbf{I} = |\det(\mathbf{B}^T\mathbf{B})| dCd\bar{R}$. In fact $p(\mathbf{I}; \mathbf{B}, \lambda) = p_C(C; \lambda) q_{\bar{R}}(\bar{R}) |\det(\mathbf{B}^T\mathbf{B})|^{-1/2}$. By the same logic as in (43) and (44), we still want to find $\mathbf{B}$ to minimize the overall reconstruction error $\sum_{m=1}^{M} \|\mathbf{I}_m - \mathbf{B}C_m\|^2$.

Under the shared sparse coding model, it is tempting to model the coefficients C of the selected basis functions directly. However, C is still a multi-dimensional vector, and direct modeling of C can be difficult. One may assume that the components of C are statistically independent for simplicity, but this assumption is unlikely to be realistic. So after selecting the basis functions, we choose to model the image intensities by the inhomogeneous FRAME model. Even though this model only matches the marginal distributions of filter responses of the selected basis functions, the model does not assume that the responses are independent.

References

1. D. H. Ackley, G. E. Hinton, T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**, 147–169. 1985. 2
2. A. Adler, M. Elad, and Y. Hel-Or. Probabilistic Subspace Clustering via Sparse Representations, *IEEE Signal Processing Letters*, **20**, 63–66, 2013. 3
3. M. Aharon, M. Elad, and A. M. Bruckstein. The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. Signal Process.*, **54**, 4311–4322, 2006. 1
4. Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *TPAMI.*, **35**, 1798–1828, 2013. 3
5. A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, **51**, 34–81, 2009. 1
6. S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, **43**, 129–159, 2001. 8
7. J. Chen and X. Huo. Sparse representations for multiple measurements vectors (mmv) in an overcomplete dictionary. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, **4**, 257–260, 2005. 3

8. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005. 13
9. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society B*, **39**, 1–38, 1977. 11
10. S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo, *Physics Letters*, **195**, 216–222, 1987. 18
11. M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, 2010. 1
12. M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Process.*, **15**, 3736–3745, 2006. 1
13. M. Elad, P. Milanfar, and R. Rubinstein. Analysis versus synthesis in signal priors. *Inverse Problems*, **23**, 2007. 3
14. R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classication. *Journal of Machine Learning Research*, **9**, 1871–1874, 2008. 16
15. 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. 16
16. P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, **32**, 1627–1645, 2010. 16
17. V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *IJCV*, **87**, 284-303, 2010. 16
18. S. Fidler, M. Boben, and A. Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. *CVPR*, 2008. 3
19. J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, **82**, 249–266, 1987. 7, 19
20. A. Gelman and X. L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, **13**, 163–185, 1998. 5
21. S. Geman, and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Trans. PAMI*, **6**, 721–741, 1984. 5
22. S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, **60**, 707–736, 2002. 3
23. B. Gong, Y. Shi, F. Sha. and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. *CVPR*, 2012. 16, 17
24. R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: an unsupervised approach. *ICCV*, 2011. 16, 17
25. G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, Caltech, 2007. 16
26. G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, **14**, 1771–1800, 2002. 2
27. G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, 1527–1554, 2006. 2, 3
28. J. Hoffman, E. Rodner, J. Donahue, K. Saenko, and T. Darrell. Efficient learning of domain-invariant image representations. *ICLR*, 2013. 16, 17
29. Y. Hong, Z. Si, W. Hu, S. C. Zhu, and Y. N. Wu, Unsupervised learning of compositional sparse code for natural image representation. *Quarterly of Applied Mathematics*, **72**, 373–406, 2013. 3, 13, 15
30. I. Jhou, D. Liu, D. T. Lee, and S. Chang. Robust visual domain adaptation with low-rank reconstruction. *CVPR*, 2012. 16, 17
31. S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006. 16
32. H. Lee, R. Grosse, R Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *ICML*, 2009. 3
33. C. Liu, S.-C. Zhu, and H.-Y. Shum. Learning Inhomogeneous Gibbs Model of Faces by Minimax Entropy. *ICCV*, 281-287, 2001. 2
34. K. Lounici, A. B. Tsybakov, M. Pontil, and S. A. van de Geer. Taking advantage of sparsity in multi-task learning. *Proceedings of the 22nd Conference on Learning Theory*, 2009 3
35. D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, **60**, 91–110, 2004. 16
36. S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, **41**, 3397–3415, 1993. 8
37. M. Marszalek and C. Schmid. Accurate object localization with shape masks. *CVPR*, 2007. 16
38. S. Nama, M. E. Daviesb, M. Eladc, R. Gribonval. The cosparse analysis model and algorithms. *Applied and Computational Harmonic Analysis*, **34**, 30-56, 2013. 3
39. R. Neal. Annealed importance sampling. *Statistics and Computing*, **11**, 125–139, 2001. 5
40. R. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2011. 5
41. G. Obozinski, M. J. Wainwright, and M. I. Jordan, Support union recovery in high-dimensional multivariate regression, *Annals of Statistics*, **39**, 1–47, 2011. 3, 8
42. 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. 1
43. Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, *The 27th Asilomar Conference on Signals, Systems and Computers*, 40–44, 1993. 8
44. S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *TPAMI*, **19**, 380–393, 1997. 19
45. M. Ranzato and G. E. Hinton. Modeling pixel means and covariances using factorized third-order Boltzmann machines. *CVPR*, 2010. 2
46. M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, **2**, 1019–1025, 1999. 8
47. S. Roth and M. Black. Fields of experts. *IJCV*, **82**, 205–229, 2009. 2
48. R. Rubinstein, M. Zibulevsky, and M. Elad, Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation, *IEEE Transactions on Signal Processing*, **58**, 1553–1564, 2010. 3
49. K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *ECCV*, 2010. 16, 17
50. S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa. Generalized domain adaptive dictionaries. *CVPR*, 2013. 16, 17
51. Z. Si and S. C. Zhu. Learning Hybrid image Template (HIT) by Information Projection. *IEEE Trans. PAMI*, **34**, 1354–1367, 2012. 16
52. P. Smolensky. Information processing in dynamical systems: foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986. 2
53. Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, **4**, 1235–1260, 2003. 2
54. R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B*, **58**, 267–288, 1996. 8
55. J. Tropp, A. Gilbert, and M. Straus. Algorithms for simultaneous sparse approximation. part I: Greedy pursuit. *Journal of Signal Processing*, **86**, 572–588, 2006. 3, 8
56. T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine. Unsupervised object discovery: a comparison. *IJCV*, 2009. 13
57. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000. 16
58. M. Welling, G. E. Hinton, and S. Osindero. Learning sparse topographic representations with products of student-t distributions. *NIPS*, 2003. 2
59. Y. N. Wu, Z. Si, H. Gong, and S. C. Zhu. Learning active basis model for object detection and recognition. *IJCV*, **90**, 198–235, 2010. 3, 7, 8

60. J. Xie, W. Hu, S. C. Zhu, Y. N. Wu. Learning Inhomogeneous FRAME Models for Object Patterns. *CVPR*, 2014. 3
61. M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher discrimination dictionary learning for sparse representation. *ICCV*, 2011. 1, 16, 17
62. L. Younes. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastic Reports*, **65**, 177–228, 1999. 2, 5
63. M. Zeiler, G. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. *ICCV*, 2011. 1, 3
64. L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised structure learning: hierarchical recursive composition, suspicious coincidence and competitive exclusion. *ECCV*, 2008. 3
65. S. C. Zhu and D. B. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, **2**, 259–362, 2006. 3
66. S. C. Zhu, Y. N. Wu, and D. B. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, **9**, 1627–1660, 1998. 2, 3, 4, 7, 19