Inducing Wavelets into Random Fields via Generative Boosting

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu[*]

Department of Statistics, University of California, Los Angeles, USA

Abstract

This paper proposes a learning algorithm for the random field models whose energy functions are in the form of linear combinations of rectified filter responses from subsets of wavelets selected from a given over-complete dictionary. The algorithm consists of the following two components. (1) We propose to induce the wavelets into the random field model by a generative version of the epsilon-boosting algorithm. (2) We propose to generate the synthesized images from the random field model using Gibbs sampling on the coefficients (or responses) of the selected wavelets. We show that the proposed learning and sampling algorithms are capable of generating realistic image patterns. We also evaluate our learning method on a dataset of clustering tasks to demonstrate that the models can be learned in an unsupervised setting. The learned models encode the patterns in wavelet sparse coding. Moreover, they can be mapped to the second-layer nodes of a sparsely connected convolutional neural network (CNN).

Keywords: Generative models, Markov random fields, Simultaneous sparse coding

1. Introduction and Motivations

1.1. Random field models based on wavelets

It is well known that wavelets provide sparse representations of natural images, in that each image can be represented by a linear combination of a small subset of wavelets selected from a given over-complete dictionary. We can exploit this fact to learn statistical models for various image patterns (such as object categories), so that

[*]Corresponding author. E-mail: *ywu@stat.ucla.edu*

each pattern is represented by a subset of wavelets selected from a given dictionary, while their coefficients (as well as their locations and orientations) are allowed to vary according to a certain probability distribution. Such a model can be written in the form of a Markov random field (or a Gibbs distribution), whose energy function is in the form of a linear combination of rectified filter responses from the subset of selected wavelets [24] . The model is generative in the sense that it is in the form of a probability distribution defined on the space of images, so that we can generate images by drawing samples from the probability distribution.

In this article, we propose a learning method for inducing wavelets into such random field models. It consists of the following two components. (1) We propose to select the wavelets by a generative version of the epsilon-boosting algorithm [10]. We call this process generative boosting because the gradient of the log-likelihood is computed based on Monte Carlo samples generated from the model. (2) We propose to generate synthesized images from the model using a Gibbs sampling algorithm [11] that samples the reconstruction coefficients (or the filter responses) of the selected wavelets, by exploiting the sparse coding form of the model. The proposed learning algorithm identifies important dimensions of variations and generates synthesized images by moving along these dimensions. It also gives a computational justification for sparsity as promoting efficient sampling of the resulting statistical model.

Learning process as a painting process. Figure 1 illustrates the learning process, which is similar to the way an artist paints a picture by sequentially fleshing out more and more details. (a) displays the training images. (b) displays the sequence of synthesized images generated by the learned model as more and more wavelets are induced into the random field by the generative boosting process. Each wavelet is like a stroke in the painting process. The wavelets are selected from a given dictionary of oriented and elongated Gabor wavelets at a dense collection of locations, orientations and scales. The dictionary also includes isotropic Difference of Gaussian (DoG) wavelets. (c) displays the sequence of sketch templates of the learned model where each selected Gabor wavelet is illustrated by a bar with the same location, orientation and length as the corresponding wavelet, and each selected DoG wavelet is illustrated by a circle (in each sketch template, wavelets of smaller scales appear darker.) (d) displays more

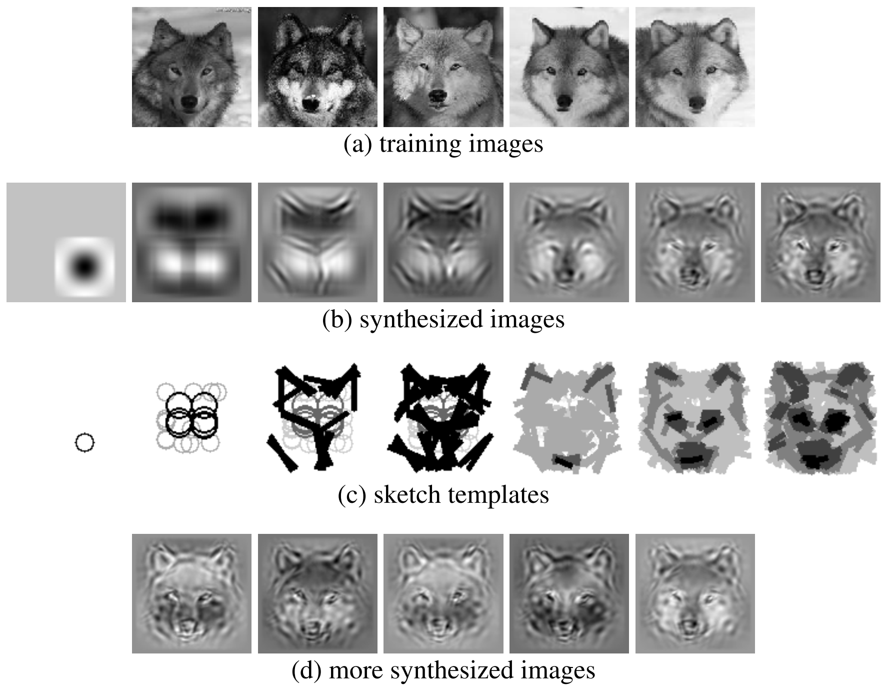


Figure 1: Learning process by generative boosting. (a) observed training images (100×100 pixels) from which the random field model is learned. (b) a sequence of synthesized images generated by the learned model as more and more wavelets are induced into the model. The numbers of the selected wavelets are $1, 20, 65, 100, 200, 500$, and 800 respectively. (c) a sequence of sketch templates that illustrate the wavelets selected from an over-complete dictionary. The dictionary includes 4 scales of Gabor wavelets, illustrated by bars of different sizes, and 2 scales of Difference of Gaussian (DoG) wavelets, illustrated by circles. In each template, smaller scale wavelets appear darker than larger ones. (d) more synthesized images independently generated from the final learned model.

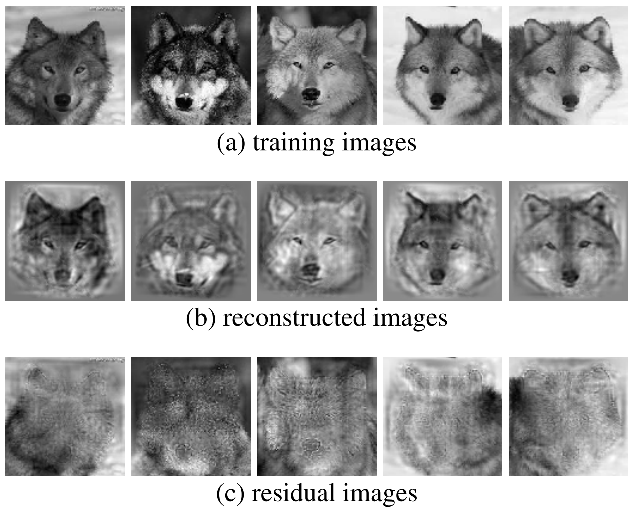


Figure 2: Shared sparse coding after wavelet selection by generative boosting. (a) observed training images. (b) images reconstructed by the selected wavelets. (c) residual images.

synthesized images generated from the final model.

Figure 1 illustrates the principle of “analysis by synthesis” or “vision = inverse graphics.” We synthesize images by sampling from the current model, and then select the next wavelet that reveals the most conspicuous difference between the synthesized images and the observed images. As more wavelets are added into the model, the synthesized images become more similar to the observed images. In other words, the generative model is endowed with the ability of imagination, which enables it to adjust its parameters so that its imaginations match the observations in terms of statistical properties that the model is concerned with.

Shared sparse coding and flexible composite basis function. The selected wavelets provide shared sparse coding of the training images, in that all the images can be encoded by the same set of selected wavelets, but with different sets of coefficients for different images (we also allow the selected wavelets to perturb their locations and orientations when encoding each image). Figure 2 illustrates the idea. (a) displays the training images. (b) displays the corresponding reconstructed images as linear superpositions of the selected wavelets. (c) displays the corresponding residual images.

We may consider the composition of the selected wavelets as a new basis function. However, this new basis function is flexible or reconfigurable in that the coefficients of the constituent wavelets (as well as their locations and orientations) are allowed to vary according to a probability distribution. If we have a dictionary of such flexible composite basis functions, it can lead to more efficient representations than the original base dictionary of wavelets.

In our experiments, we show that the proposed learning and sampling algorithms are capable of generating realistic image patterns. For numerical evaluation, we test our learning algorithm on a dataset for clustering tasks, and compare its performance with other clustering methods. The clustering experiment shows that the models can be learned in an unsupervised setting.

Having explained the basic idea, we now give two motivations for our work.

1.2. Motivation 1: high-level representations of sparse-land

Wavelet sparse coding underlies many successful methods in signal processing, such as image compression, de-noising, and super-resolution [7] [14]. During the past two decades, tremendous progress has been made in both algorithm design as well as theoretical analysis in this interdisciplinary area of research, driven by researchers from harmonic analysis, statistics and machine learning. Most of the past activities in this area are based on the so-called "sparse-land" assumption [7], that is, the signals to be recovered admit sparse linear representations by an over-complete dictionary of basis functions, and the signals can be recovered by solving an inverse problem regularized by a sparsity-inducing norm or penalty function.

However, beyond generic sparsity, little effort has been made to understand the objects in the sparse-land: What do these objects look like? Can we find higher level representations for these objects? An obvious observation is that in the sparse coding of the objects in the sparse-land, the wavelets do not enter the sparse representations randomly. Instead, they form various grouping or compositional patterns, and their coefficients may also follow certain patterns. The random field models and the generative boosting method studied in this paper seek to discover and encode such patterns, and may lead to dictionaries of flexible composite basis functions as mentioned in the above subsection.

For natural images, one can learn an over-complete dictionary of basis functions by enforcing sparsity [16]. The learned basis functions resemble elongated and oriented Gabor wavelets at different scales, and they encode ubiquitous simple structures such as edges. The random field models studied in this paper encode more complex patterns by selecting and composing wavelets, while allowing their coefficients (as well as locations and orientations) to vary for flexibility or reconfigurability. Such models can be useful for object recognition and scene understanding.

1.3. Motivation 2: generative convolutional neural nets

Recent years have witnessed tremendous successes of the convolutional neural networks (ConvNet or CNN) [12] [13]. The CNN is a hierarchical structure that consists of alternative layers of linear filtering, non-linear rectification and max-pooling (as well as sub-sampling). When trained on the imagenet dataset [6] (a large dataset of natural images), the learned filters at the lowest layer resemble Gabor wavelets, possibly due to the rectification non-linearity that encourages sparsity.

The random field models studied in this paper can be mapped to the nodes at the second layer of a CNN, but nodes corresponding to these models are sparsely and selectively connected to the first layer nodes of Gabor-like filters, so these models are more explicit and meaningful than the common CNN nodes. More importantly, these models are generative in that synthesized images can be sampled from the probability distributions defined by these models. Such models can be trained in an unsupervised setting by maximum likelihood. In terms of sparse coding discussed in the previous subsection, these models represent commonly occurring sparse coding patterns of wavelets, where sparse activities are hardwired into sparse connectivities. Understanding the learning of such models may eventually pave the way to learn sparsely connected CNN in a generative and unsupervised fashion.

2. Past work

In addition to the research themes mentioned in the Introduction section, the following are the models and methods that are closely related to our work.

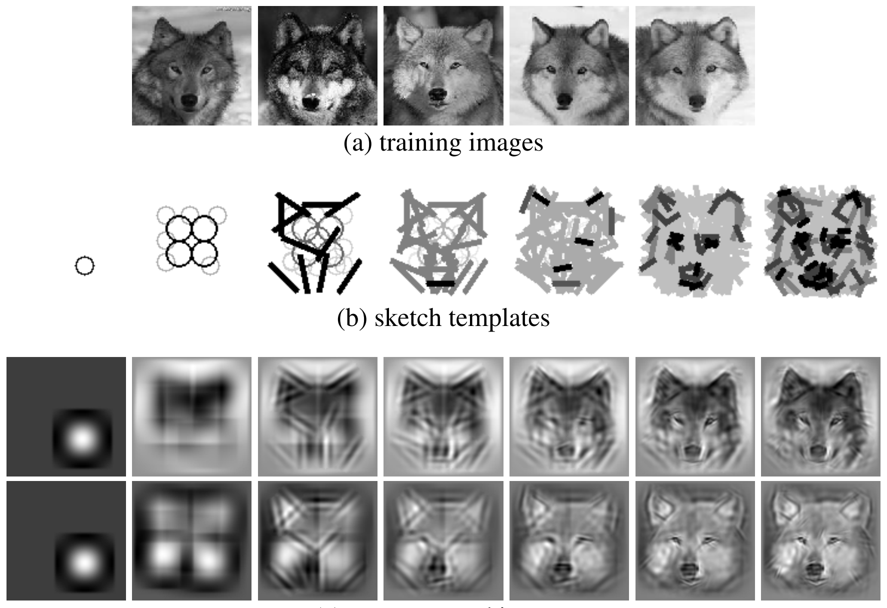


Figure 3: Shared matching pursuit for the purpose of wavelet selection. (a) training images. (b) sequence of sketch templates that illustrate the wavelets selected sequentially in order to reconstruct all the training images simultaneously. See the caption of Figure 1(c) for the explanation of sketch templates. The selected wavelets are shared by all the training images in their reconstructions. The numbers of selected wavelets in the sequence are 2, 20, 60, 100, 200, 500, and 800 respectively. (c) sequences of reconstructed images by the selected wavelets for the 1st and 3rd training images in (a).

2.1. Sparse FRAME model and two-stage learning

The random field model based on wavelets is called the sparse FRAME model in our previous work [24]. The model is an inhomogeneous and sparsified generalization of the original FRAME (Filters, Random field, And Maximum Entropy) model [27]. In [24] , we developed a two-stage learning algorithm for the sparse FRAME model. The algorithm consists of the following two stages. Stage 1: selecting a subset of wavelets from a given dictionary by a shared sparse coding algorithm, such as shared matching pursuit. Stage 2: estimating the parameters of the model by maximum likelihood via stochastic gradient ascent where the MCMC sampling is powered by Hamiltonian Monte Carlo [15]. Figure 3 illustrates the shared matching pursuit algorithm. Again (a) displays the training images. (b) displays the sequence of sketch templates that illustrate the selected wavelets. (c) displays the sequences of reconstructed images for 2 of the 5 training images as more and more wavelets are selected by the shared matching pursuit.

One conceptual concern with the above two-stage learning algorithm is that the selection of the wavelets in the first stage is not guided by the log-likelihood, but by a least squares criterion that is only an approximation to the log-likelihood. In other words, the first stage seeks to reconstruct the training images by selecting wavelets, without any regard for the patterns formed by the coefficients of the selected wavelets.

This paper is a continuation of our previous work on the sparse FRAME model [24]. Relative to [24], the main contribution of this paper is that we replace the computational core of the original sparse FRAME model with new learning and sampling algorithms. In the learning algorithm, both the selection of the wavelets and the estimation of the parameters are guided by the log-likelihood, so the learning algorithm is a more elegant one-stage algorithm in contrast to the two-stage algorithm in our previous work [24]. Moreover, in this paper, the sampling of synthesized images is accomplished by Gibbs sampling that moves the images along the dimensions spanned by the selected wavelets. We show that Gibbs sampling is less prone to local modes than the Hamiltonian Monte Carlo sampler employed in [24].

2.2. Simultaneous or shared sparse coding

The random field model studied in this paper can be written in the form of simultaneous sparse coding [2] [19] or shared sparse coding as it was called in our previous work [24], where the training images can be encoded by a common set of wavelets selected from a given dictionary, and the coefficients of the selected wavelets follow a well-defined probability distribution. In fact, in our previous work [24], we employed simultaneous or shared sparse coding to select the wavelets before estimating the parameters of the probability distribution of the image. See Figure 3 for an illustration. Compared to our work, the existing methods on simultaneous sparse coding in harmonic analysis literature did not aim to construct probability models for the wavelet coefficients. As a consequence, they can reconstruct the input images but can-not synthesize new images as in our work. From the harmonic analysis perspective, simultaneous sparse coding may be the entry point to high-level representations of the sparse-land.

2.3. Projection pursuit

The generative boosting method studied in this paper can be considered a parametric and finite version of projection pursuit [9], where the selection of the wavelets is guided by the log-likelihood of a parametric model, and the wavelets are selected from a finite over-complete dictionary instead of the infinite set of all possible projections. In this sense, our work provides a practical implementation and a vivid illustration of projection pursuit.

2.4. Epsilon-Boosting

The epsilon-boosting algorithm proposed by Friedman [10] was inspired by adaboost [8]. Its relation with the solution path of the ℓ_1 regularized loss function was studied by [17]. Epsilon-boosting has mainly been used in discriminative learning, such as learning boosting tree classifiers [10].

2.5. Generative boosting

The literature on the application of boosting in learning generative models is relatively scarce. The seminal paper [5] proposed a pursuit method for inducing features into random fields. Our work can be considered a highly specialized example with a very different and specialized computational core. Our work can also be considered a variation of the filter pursuit method of the original FRAME paper [27]. The difference from [27] is that in our work the selected wavelets are location specific and are intended for representing object patterns instead of texture patterns.

[22] proposed to use boosting methods to learn a generalized version of the restricted Boltzmann machine on binary data. They used an approximate MCMC method for sampling from the current model. In contrast, we develop a Gibbs sampling method to sample the coefficients of the selected wavelets. Also, [22] was not intended for learning sparse models or for performing sparse selection from a given dictionary. [20]

proposed an interesting theory for combining adaboost [8] and generative sampling for learning binary features, and demonstrated its applications in various image processing tasks. The features that we use are not binary, and our sampling and learning methods are different from [20].

3. Random fields in sparse-land

The random field models studied in this paper are in the form of Gibbs distributions whose energy functions are in the form of linear combinations of rectified filter responses from a set of wavelets. They are called inhomogeneous FRAME (Filters, Random field, And Maximum Entropy) models in [24] because they are an inhomogeneous generalization of the original FRAME model [27]. We first describe the dense version of the inhomogeneous FRAME model where all the wavelets in the dictionary are included. We then describe the sparsified version of the inhomogeneous FRAME model (called sparse FRAME model in [24]) where only a subset of wavelets is selected from the dictionary.

3.1. Dense random field

Following the notation of the inhomogeneous FRAME model [24], let $\mathbf{I}$ be an image defined on the rectangular domain $\mathcal{D}$. Let $B_{x,s,\alpha}$ denote a basis function such as a Gabor wavelet (or difference of Gaussian (DoG) filter) centered at pixel x (a two-dimensional vector) and tuned to scale s and orientation α. We discretize both s and α so that they take values in finite sets. In our current implementation, α is discretized into 16 equally spaced orientations. Given a dictionary of wavelets or filter bank $\{B_{x,s,\alpha}, \forall x, s, \alpha\}$, the dense version of the random field model is of the following form

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

which is a Markov random field model because the $B_{x,s,\alpha}$ are localized. The model is also a Gibbs distribution, an exponential family model, a log-linear model, or an energy-based model, as it is called in different contexts.

In model (1), $\langle \mathbf{I}, B_{x,s,\alpha} \rangle$ is the inner product between $\mathbf{I}$ and $B_{x,s,\alpha}$, and it is called the filter response of $\mathbf{I}$ to the wavelet $B_{x,s,\alpha}$. $h()$ is a non-linear rectification function, such as a piecewise linear function. In this paper, we adopt the choice $h(r) = |r|$, which is sometimes called full-wave rectification. Another choice is $h(r) = \max(0, r)$, which is sometimes called half-wave rectification. A more elaborate version is $h(r) = \max(0, r - t)$ where t is a threshold. This $h()$ is also called the rectified linear unit in the machine learning literature [12].

In (1), $\lambda = (\lambda_{x,s,\alpha}, \forall x, s, \alpha)$ are the weight parameters to be estimated from the training images. $Z(\lambda)$ is the normalizing constant. $q(\mathbf{I})$ is a white noise reference distribution whose pixel values follow independent $\mathrm{N}(0, \sigma^2)$ distributions, so

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

where $||\mathbf{I}||$ is the ℓ_2 norm of $\mathbf{I}$, and $|\mathcal{D}|$ denotes the number of pixels in the image domain $\mathcal{D}$. In this paper, we fix $\sigma^2 = 1$. We normalize the training images to have marginal mean 0 and a fixed marginal variance (e.g., 10). For the choice $h(r) = |r|$, if $\lambda_{x,s,\alpha} > 0$, then the model encourages large magnitude of the filter response $\langle \mathbf{I}, B_{x,s,\alpha} \rangle$, regardless of its sign. This is often appropriate if we want to model object shape patterns.

Maximum likelihood learning by stochastic gradient. The basic learning algorithm estimates the unknown parameters λ from a set of aligned training images $\{\mathbf{I}_m, m = 1, ..., M\}$ that come from the same object category, where M is the total number of training images. This basic algorithm can then be generalized to learn multiple models from non-aligned images sampled from multiple categories by an EM-like algorithm. It can be further generalized to learn a codebook of models from non-aligned images. In the basic learning algorithm, the weight parameters λ can be estimated by maximizing the log-likelihood function

$$L(\lambda) = \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{I}_m; \lambda), \tag{3}$$

whose partial derivatives are

$$\frac{\partial L(\lambda)}{\partial \lambda_{x,s,\alpha}} = \frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle| - \mathrm{E}_\lambda \left[|\langle \mathbf{I}, B_{x,s,\alpha} \rangle| \right], \tag{4}$$

where E_λ denotes expectation with respect to $p(\mathbf{I};\lambda)$, and it can be approximated by Monte Carlo integration. Thus λ can be computed by the stochastic gradient ascent algorithm [25]

$$\lambda_{x,s,\alpha}^{(t+1)} = \lambda_{x,s,\alpha}^{(t)} + \gamma_t \left(\frac{1}{M} \sum_{m=1}^{M} |\langle \mathbf{I}_m, B_{x,s,\alpha} \rangle| - \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle| \right), \tag{5}$$

where γ_t is the step size or learning rate, $\{\tilde{\mathbf{I}}_m\}$ are the synthesized images sampled from $p(\mathbf{I};\lambda^{(t)})$ using MCMC. $\tilde{M}$ is the total number of independent parallel Markov chains that sample from $p(\mathbf{I};\lambda^{(t)})$. The parameter learning follows the "analysis by synthesis" principle, which iteratively synthesizes images from the current model and updates the weight parameters until the statistical properties of the synthesized images match those of the observed images.

Writing the model as $p(\mathbf{I};\lambda) \propto \exp(-U(\mathbf{I}))$, where the energy function

$$U(\mathbf{I}) = -\sum_{x,s,\alpha} \lambda_{x,s,\alpha} |\langle \mathbf{I}, B_{x,s,\alpha} \rangle| + \frac{1}{2\sigma^2} \|\mathbf{I}\|^2, \tag{6}$$

the Hamiltonian Monte Carlo (HMC) algorithm [15] is used by [24] to sample from $p(\mathbf{I};\lambda)$, which involves computing $U'(\mathbf{I})$.

Normalizing constant. The learning of the model also involves the estimation of the normalizing constant, which plays an important role in fitting the mixture models or in unsupervised learning, even though it is not required for estimating λ and synthesizing images. The ratio of the normalizing constants at two consecutive steps is approximated by

$$\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})} \approx \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} \left[\exp\Big(\sum_{x,s,\alpha} (\lambda_{x,s,\alpha}^{(t+1)} - \lambda_{x,s,\alpha}^{(t)}) \times |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle| \Big) \right]. \tag{7}$$

Starting form $\lambda^{(0)} = 0$ and $\log Z(\lambda^{(0)}) = 0$, $\log Z(\lambda^{(t)})$ can be computed along the learning process.

3.2. Sparse random field model for images in sparse-land

The dense random field model (1) can be sparsified so that only a small number of $\lambda_{x,s,\alpha}$ are non-zero, i.e., only a small subset of wavelets is selected from the given dictionary. This leads to the following sparse random field model [24]:

$$p(\mathbf{I}; \mathbf{B}, \lambda) = \frac{1}{Z(\lambda)} \exp\left(\sum_{i=1}^{n} \lambda_i |\langle \mathbf{I}, B_{x_i,s_i,\alpha_i}\rangle|\right) q(\mathbf{I}), \tag{8}$$

where $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ are the n wavelets selected from the given dictionary, and $\lambda = (\lambda_i, i = 1, ..., n)$ are the corresponding weight parameters. Random fields like (8) generate images that belong to the sparse-land. This point will be made explicit in Section 5.1.

Writing the model as $p(\mathbf{I}; \mathbf{B}, \lambda) \propto \exp(-U(\mathbf{I}))$, then

$$U(\mathbf{I}) = -\sum_{i=1}^{n} \lambda_i |\langle \mathbf{I}, B_{x_i,s_i,\alpha_i}\rangle| + \frac{1}{2\sigma^2}\|\mathbf{I}\|^2. \tag{9}$$

$U(\mathbf{I})$ is piecewise quadratic, or the model is piecewise Gaussian. There are 2^n pieces according to the sign patterns of the responses $(\langle \mathbf{I}, B_{x_i,s_i,\alpha_i}\rangle, i = 1, ..., n)$.

In the above model, n is assumed given (e.g., $n = 200$), although it can be chosen adaptively by some information criteria. The dictionary of wavelets is also assumed given. Otherwise it can be learned from training images by sparse coding [16]. The selected wavelets can also be re-learned as unknown parameters of the model by maximum likelihood via stochastic gradient.

4. Generative boosting for inducing wavelets into random field

For notational simplicity, we use the form of the dense model in equation (1), but we assume $\lambda = (\lambda_{x,s,\alpha}, \forall x, s, \alpha)$ is a sparse vector, i.e., only a small number of $\lambda_{x,s,\alpha}$ are non-zero. In the context of variable selection, the epsilon-boosting algorithm [10] is similar to coordinate ascent except that it only takes a small step size (thus the word "epsilon" in "epsilon-boosting") to update the coordinate. In fitting the model (1), the algorithm starts from $\lambda = 0$, the zero vector. Then the generative boosting algorithm updates the weight parameters λ based on the Monte Carlo estimation of the gradient of the log-likelihood, $L'(\lambda)$. However, unlike the updating scheme in equations (5) where all the weight parameters are adjusted according to the gradient, generative epsilon-boosting only chooses to update the particular weight parameter that corresponds to the maximal coordinate of the gradient.

Specifically, at the t-th step, let

$$D(x, s, \alpha) = \frac{1}{M} \sum_{m=1}^{M} \max_{\Delta x, \Delta \alpha} |\langle \mathbf{I}_m, B_{x+\Delta x, s, \alpha+\Delta\alpha} \rangle| - \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle|, \quad (10)$$

where $(\Delta x, \Delta \alpha)$ are the perturbations of the location and orientation of the basis function $B_{x,s,\alpha}$. The perturbations are introduced to account for shape deformations in the observed images, and they can take values within some small ranges. In our current implementation, the magnitude of Δx is up to 2 or 3 pixels, and $\Delta\alpha$ is within $\{-\pi/16, 0, \pi/16\}$. The local max pooling is only applied to the observed images to filter out shape deformations, and we assume $p(\mathbf{I}; \lambda)$ models the images prior to shape deformations. So $p(\mathbf{I}; \lambda)$ is a deformable template, and there is an explicit separation between appearance and shape variations in the model.

We select

$$(\hat{x}, \hat{s}, \hat{\alpha}) = \arg\max_{x,s,\alpha} D(x, s, \alpha), \quad (11)$$

and update $\lambda_{\hat{x},\hat{s},\hat{\alpha}}$ by

$$\lambda_{\hat{x},\hat{s},\hat{\alpha}} \leftarrow \lambda_{\hat{x},\hat{s},\hat{\alpha}} + \gamma_t D(\hat{x}, \hat{s}, \hat{\alpha}), \quad (12)$$

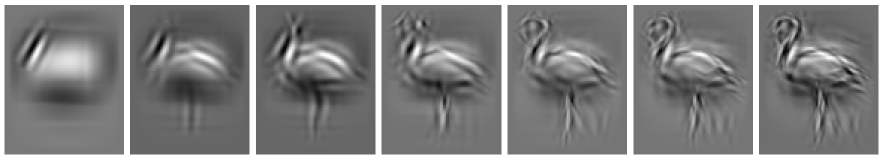
where γ_t is the step size, assumed to be a sufficiently small value. The selected wavelet $B_{\hat{x},\hat{s},\hat{\alpha}}$ reveals the dimension along which the current model is most conspicuously lacking in reproducing the statistical properties of the training images. By including $B_{\hat{x},\hat{s},\hat{\alpha}}$ into the model and updating the corresponding parameter $\lambda_{\hat{x},\hat{s},\hat{\alpha}}$, the model receives the most needed boost. In terms of the painting analogy, $B_{\hat{x},\hat{s},\hat{\alpha}}$ is the stroke that is most needed to make the painting look more similar to the observed objects.

The epsilon-boosting algorithm has an interesting relationship with the ℓ_1 regularization in the Lasso [18] and basis pursuit [3]. As pointed out by [17], under a monotonicity condition (e.g., the components of λ keep increasing), such an algorithm approximately traces the solution path of the ℓ_1 regularized minimization problem:

$$-\frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{I}_m; \lambda) + \rho \|\lambda\|_{\ell_1}, \quad (13)$$

where the regularization parameter ρ starts from a big value so that all the components

(a) observed images


(b) synthesized images

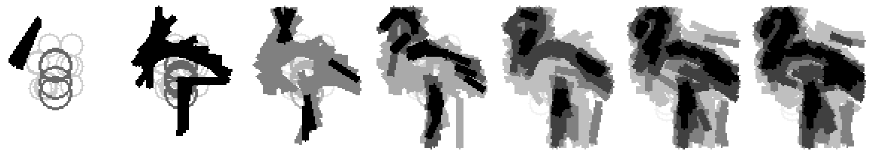

(c) sketch templates

Figure 4: Learning sequence by generative epsilon-boosting. (a) some of the training images. (b) synthesized images (100×80) are generated when the number of selected wavelets = 20, 60, 100, 200, 400, 600, and 700. (c) corresponding sketch templates.

of λ are zero, and gradually lowers itself to allow more components to be non-zero so that more wavelets are induced into the model.

Algorithm (1) gives the details of the algorithm. In parameter updating, we update the weight parameters of all the selected wavelets for the sake of computational efficiency. It is possible that a wavelet can be selected more than once in the boosting process (as well as the shared matching pursuit process in [24]), so the actual number of distinct wavelets selected can be slightly less than the number of selected wavelets (or more precisely the number of selection operations) reported in the paper.

Figures 4 and 5 illustrate the learning process by displaying the synthesized images generated by the learned model as more and more wavelets are included. The corresponding sketch templates are also displayed.

Shared sparse coding. The observed images $\{\mathbf{I}_m, \forall m\}$ can be encoded using the

Algorithm 1 Learning algorithm

Input:

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

(2) number of wavelets to be selected n

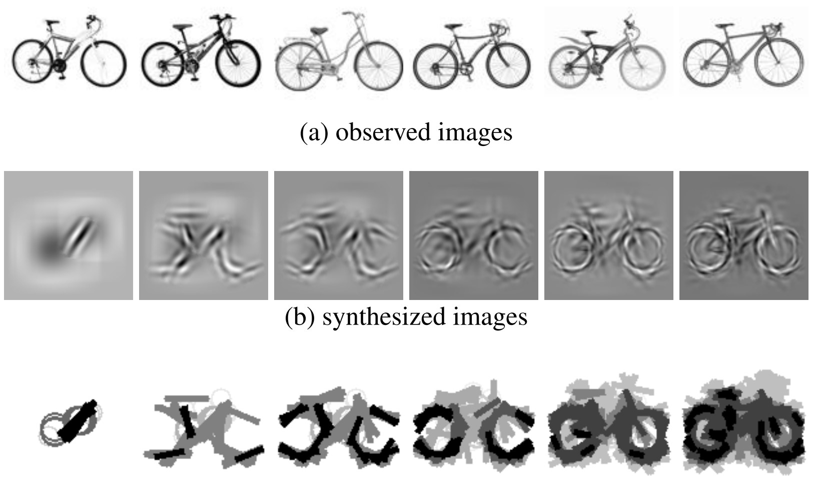
Output:

(1) set of selected wavelets $\mathbf{B} = \{B_{x_i, s_i, \alpha_i}, i = 1, ..., n\}$

(2) set of associated parameters $\lambda = \{\lambda_i, i = 1, ..., n\}$

(3) logarithm of normalizing constant $\log Z(\lambda)$.

```
1:  Let $\mathbf{B} \leftarrow \phi$, $\lambda \leftarrow 0$, $\log Z(\lambda^{(0)}) = 0$, and $t = 0$
2:  Initialize $\tilde{\mathbf{I}}_m$ as Gaussian white noise images
3:  Calculate observed statistics:
    $H^{obs}_{x,s,\alpha} \leftarrow \frac{1}{M} \sum_{m=1}^{M} \max_{\Delta x, \Delta \alpha} |\langle \mathbf{I}_m, B_{x+\Delta x, s, \alpha+\Delta\alpha} \rangle|, \forall x, s, \alpha$
4:  **repeat**
5:      Generate $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ from $p(\mathbf{I}; \mathbf{B}^{(t)}, \lambda^{(t)})$ by the Gibbs sampler in
        Algorithm (2)
6:      Calculate synthesized statistics:
        $H^{syn}_{x,s,\alpha} \leftarrow \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} |\langle \tilde{\mathbf{I}}_m, B_{x,s,\alpha} \rangle|, \forall x, s, \alpha$
7:      Select $(\hat{x}, \hat{s}, \hat{\alpha}) = \arg\max_{x,s,\alpha} (H^{obs}_{x,s,\alpha} - H^{syn}_{x,s,\alpha})$
8:      $\mathbf{B}^{(t+1)} \leftarrow \mathbf{B}^{(t)} \cup \{B_{\hat{x},\hat{s},\hat{\alpha}}\}$
9:      Update $\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \gamma_t (H_i^{obs} - H_i^{syn})$, $i = 1, ..., |\mathbf{B}^{(t+1)}|$.
10:     Compute $Z$ ratio $\frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$ by Eq. (7)
11:     Update $\log Z(\lambda^{(t+1)}) \leftarrow \log Z(\lambda^{(t)}) + \log \frac{Z(\lambda^{(t+1)})}{Z(\lambda^{(t)})}$
12:     $t \leftarrow t + 1$
13: **until** $|\mathbf{B}^{(t)}| = n$ ($|\mathbf{B}|$ denotes the number of distinct wavelets in $\mathbf{B}$)
```


(a) observed images

(b) synthesized images

(c) sketch templates

Figure 5: (a) some of the training images. (b) synthesized images (100×100) are generated when the number of selected wavelets = 20, 60, 100, 200, 400, and 700. (c) sketch templates.

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

$$\mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}} + \epsilon_m, \tag{14}$$

where $(\Delta x_{m,i}, \Delta\alpha_{m,i})$ are the perturbations of the location and orientation of the i-th wavelets B_{x_i,s_i,α_i} on the m-th training image $\mathbf{I}_m$. $c_{m,i}$ are the coefficients of the selected wavelets for encoding $\mathbf{I}_m$. We call (14) shared sparse coding because the selected wavelets are shared by all the observed images. See Figure 2 in Section 1.1 for an illustration.

Correspondence to CNN. The filer bank $\{B_{x,s,\alpha}, \forall x, s, \alpha\}$ can be mapped to the first layer of Gabor-like filters of a CNN. The $\max_{\Delta x, \Delta\alpha}$ operations correspond to the max pooling layer. Each sparse random field model (8) can be mapped to a node in the second layer of a CNN, where each second layer node is sparsely and selectively connected to the nodes of the first layer, with λ_i being the connection weights. The generative boosting algorithm is used to sparsely wire the connections. $-\log Z(\lambda)$ can be mapped to the bias term of the second-layer node.

5. Gibbs sampler on wavelets coefficients

In order to power the generative boosting algorithm, we need to sample from the currently learned model. In this section, we develop a Gibbs sampling algorithm that generates the synthesized images by exploiting the generative form of the model.

5.1. Wavelet sparse coding

The sparse model in equation (8) is in the form of a Markov random field model where the selected wavelets serve as filters. It can also be translated into a generative model where the selected wavelets serve as linear basis functions. Specifically, given the selected wavelets or basis functions $\mathbf{B} = (B_i = B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$, we can represent the image $\mathbf{I}$ by

$$\mathbf{I} = \sum_{i=1}^{n} c_i B_{x_i,s_i,\alpha_i} + \epsilon, \tag{15}$$

where c_i are the least squares reconstruction coefficients, and ϵ is the residual image after we project $\mathbf{I}$ onto the subspace spanned by $\mathbf{B}$. Note that $C = (c_i, i = 1, ..., n)$ is a deterministic transformation of $\mathbf{I}$ given $\mathbf{B}$. Specifically, we can write $\mathbf{I}$, B_i, as well as C as column vectors, so that $\mathbf{B} = (B_i, i = 1, ..., n)$ is a matrix. Then C is the least squares regression coefficients of $\mathbf{I}$ on $\mathbf{B}$, i.e., $C = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{I}$.

It can be shown that under the model $p(\mathbf{I}; \mathbf{B}, \lambda)$ in (8), the distribution of C is

$$p_C(C; \lambda) = \frac{1}{Z(\lambda)} \exp\left(\langle\lambda, |\mathbf{B}^\top\mathbf{B}C|\rangle\right) q_C(C), \tag{16}$$

where for a vector v, the notation $|v|$ denotes the vector obtained by taking the absolute value of each element of v. In equation (16),

$$q_C(C) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(C^\top(\mathbf{B}^\top\mathbf{B})C)\right) |\mathbf{B}^\top\mathbf{B}|^{1/2} \tag{17}$$

is a multivariate Gaussian distribution, i.e., under $q_C(C)$, $C \sim \mathrm{N}(0, \sigma^2(\mathbf{B}^\top\mathbf{B})^{-1})$. Moreover, under $p(\mathbf{I}; \mathbf{B}, \lambda)$ in (8), C and ϵ are independent of each other. Specifically, ϵ is the Gaussian white noise image (following $q(\mathbf{I})$) subtracted by its least squares projection on $\mathbf{B}$.

The appendix gives a detailed derivation of $p_C(C)$. The basic idea is quite simple. Under $q(\mathbf{I})$, C and ϵ are independent of each other since they are obtained by the projections of $\mathbf{I}$ on $\mathbf{B}$ and the remaining dimensions that are orthogonal to $\mathbf{B}$ respectively. The change from $q(\mathbf{I})$ to $p(\mathbf{I}; \mathbf{B}, \lambda)$ only involves the projection of $\mathbf{I}$ on $\mathbf{B}$, thus under $p(\mathbf{I}; \mathbf{B}, \lambda)$, only the distribution of C is changed from $q_C(C)$, which is $\mathrm{N}(0, \sigma^2(\mathbf{B}^\top \mathbf{B})^{-1})$, to $p_C(C; \lambda)$, while the distribution of ϵ remains the same as that under $q(\mathbf{I})$, and C and ϵ remain independent.

Thus in order to generate an image from $p(\mathbf{I}; \mathbf{B}, \lambda)$, we can generate C from $p_C(C; \lambda)$ using the Gibbs sampler. Then we get the synthesized image according to (15). To generate ϵ, we can first generate each of its component from $\mathrm{N}(0, \sigma^2)$ independently. Then we project ϵ onto $\mathbf{B}$, and take the remainder as the noise term ϵ to be added to $\mathbf{B}C = \sum_{i=1}^n c_i B_i$ in (15). We can also just keep $\mathbf{I} = \sum_{i=1}^n c_i B_i$ as the synthesized image while discarding the noise ϵ. In this article, the synthesized images that we show are such noiseless versions. We can use the noiseless synthesized images for learning λ or selecting the next wavelet in generative boosting.

The distribution $p_C(C)$ is also a Markov random field. c_i and c_j ($i \neq j$) are neighbors if the corresponding wavelets B_i and B_j have non-zero inner product $\langle B_i, B_j \rangle$. The conditional distribution of c_i given all the other components, denoted as c_{-i}, only depends on those c_j that are neighbors of c_i.

Note that if $\mathbf{B}^\top \mathbf{B} = I_n$, i.e., the selected wavelets are orthogonal, then $c_i = \langle \mathbf{I}, B_i \rangle$, and all the components of C are independent of each other. If the selected wavelets are not heavily correlated, then the coefficients are weakly dependent, and the Gibbs sampling of C is expected to be fast mixing.

5.2. Moving along basis vectors

For Gibbs sampling of $C \sim p_C(C; \lambda)$, an equivalent but simpler way of updating the i-th component is by first sampling from $p(d_i) \propto p(\mathbf{I} + d_i B_i; \mathbf{B}, \lambda)$, and letting $c_i \leftarrow c_i + d_i$, and $\mathbf{I} \leftarrow \mathbf{I} + d_i B_i$, where d_i is a scalar increment of c_i. The change from $\mathbf{I}$ to $\mathbf{I} + d_i B_i$ is a move along the basis vector B_i, and it only changes c_i while keeping the other coefficients fixed. This is because $(c_i, i = 1, ..., n)$ are least squares reconstruction coefficients, so the $d_i B_i$ term will be perfectly accounted for by adding d_i to c_i without changing the least squares reconstruction error. So this scheme implements the same move as an update in the Gibbs sampler. See the appendix for a

Algorithm 2 Sampling algorithm

```
Input:
    (1) set of selected wavelets $\mathbf{B} = \{B_{x_i,s_i,\alpha_i}, i = 1, ..., n\}$
    (2) set of parameters $\lambda = \{\lambda_i, i = 1, ..., n\}$
    (3) number of sweeps $K$
Output:
    a synthesized image $\tilde{\mathbf{I}}$ from $p(\mathbf{I}; \mathbf{B}, \lambda)$

1: Initialize $\tilde{\mathbf{I}}$ as Gaussian white noise image
2: repeat
3:     Randomly pick $B_i$
4:     Sample $d_i$ from $p(\tilde{\mathbf{I}} + d_i B_i)$
5:     Update $\tilde{\mathbf{I}} \leftarrow \tilde{\mathbf{I}} + d_i B_i$
6: until $K$ sweeps (each sweep has $n$ updates)
```

detailed justification of the equivalence between this scheme and the Gibbs sampler.

The algorithm is presented in Algorithm (2). In sampling d_i, we discretize it into a finite number of equally spaced values. Such a scheme was used in [26] for the original FRAME model, but their method does not correspond to a Gibbs sampler on wavelet coefficients. We maintain $\tilde{M}$ parallel chains for sampling $\tilde{\mathbf{I}}_m$, $m = 1, ..., \tilde{M}$.

The sampling algorithm is a natural match to the learning algorithm. Each iteration of the generative boosting learning algorithm selects a wavelet and updates the associated weight parameter. The sampling algorithm then moves the image along the selected dimension as well as existing dimensions.

5.3. Gibbs sampling of filter responses

The Gibbs sampling can also be applied to the filter responses ($r_i = \langle \mathbf{I}, B_{x_i,s_i,\alpha_i} \rangle, i = 1, ..., n$). Let $R = (r_i, i = 1, ..., n)$. Writing $\mathbf{I}$, B_i and R as column vectors, it follows that $R = \mathbf{B}^\top \mathbf{I}$. The distribution of R under $q(\mathbf{I})$ is $q_R(R)$, which is $\mathrm{N}(0, \sigma^2 \mathbf{B}^\top \mathbf{B})$. Under the change from $q(\mathbf{I})$ to $p(\mathbf{I}; \mathbf{B}, \lambda)$, the distribution of R is changed from $q_R(R)$

to $p_R(R; \lambda)$, which is

$$p_R(R; \lambda) = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |R| \rangle - \frac{1}{2\sigma^2} R^\top (\mathbf{B}^\top \mathbf{B})^{-1} R \right). \quad (18)$$

The above is a piecewise Gaussian distribution. There are 2^n pieces, corresponding to the 2^n combinations of the signs of $(r_i, i = 1, ..., n)$. Let $A = (a_{ij})_{n\times n} = (\mathbf{B}^\top \mathbf{B})^{-1}$. The conditional distribution of r_i given the other components, denoted as r_{-i}, is

$$p_R(r_i | r_{-i}) \propto \exp\left(\lambda_i |r_i| - \frac{1}{2\sigma^2} \big(a_{ii} r_i^2 + 2 r_i \sum_{j\neq i} a_{ij} r_j \big) \right), \quad (19)$$

which is piecewise Gaussian and consists of a positive piece and a negative piece. The appendix gives details on how to sample from this piecewise Gaussian distribution. After sampling R, the synthesized image can be obtained by least squares reconstruction $\mathbf{I} = \mathbf{B}C$, where the least squares reconstruction coefficients $C = (\mathbf{B}^\top \mathbf{B})^{-1} R$. In our current Matlab implementation, with 500 wavelet selected, it takes less than 0.2 second for one sweep of the Gibbs sampling of filter responses for a total of 36 parallel chains.

In Gibbs sampling of the reconstruction coefficients or filter responses, we update one coefficient or one response at each step. It is possible to update multiple coefficients or responses of highly correlated wavelets together, albeit at a higher computational cost.

6. Experiments

6.1. Qualitative experiments

Figure 6 displays some images generated by the sparse random field models, which are learned by the generative epsilon-boosting algorithm from roughly aligned images. The proposed learning algorithm is capable of modeling a wide variety of image patterns, including highly textured patterns such as cheetah and falcon. The dictionary consists of 4 scales of Gabor wavelets and 2 scales of Difference of Gaussian (DoG) wavelets. We use 36 parallel chains to sample images from the learned models.

6.2. Mixing of sampling algorithm

We evaluate the mixing of the Gibbs sampling on wavelet coefficients by running $\tilde{M} = 100$ parallel chains that sample from the fitted model $p(\mathbf{I}; \mathbf{B}, \lambda)$ in (8) learned

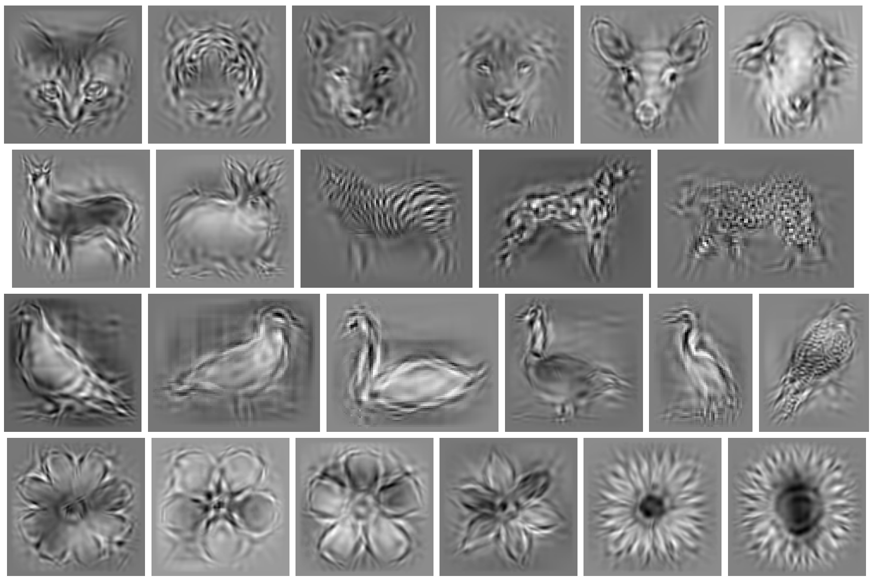


Figure 6: Images generated by the sparse random field models learned from images of different categories of objects. Typical sizes of the images are 100×100. Typical number of training images for each category is around 5. Number of the selected wavelets is 700.

from cat images, where $\mathbf{B} = (B_i, i = 1, ..., n)$ are the selected wavelets. These chains start from independent white noise images. Let $\tilde{\mathbf{I}}_{m,t}$ be the image of chain m produced at iteration t, where each iteration is a sweep of the Gibbs sampler with random scan. Let $r_{m,t,i} = \langle \tilde{\mathbf{I}}_{m,t}, B_i \rangle$ be the response of the synthesized image $\tilde{\mathbf{I}}_{m,t}$ to B_i. Let $R_{t,i} = (r_{m,t,i}, m = 1, ..., \tilde{M})$ be the $\tilde{M}$ dimensional vector. Fix $t = 100$, let $\rho_{k,i}$ be the correlation between vectors $R_{t,i}$ and $R_{t+k,i}$. Then $\rho_k = \sum_{i=1}^{n} \rho_{k,i}/n$ measures the average auto-correlation of lag k, and is an indicator of how well the parallel chains are mixing. As a comparison, we also compute ρ_k for the Hamiltonian Monte Carlo (HMC) sampler employed in our previous work [24], where each iteration consists of 30 leapfrog steps. Figure 7 plots ρ_k for $k = 1, ..., 20$, and for $n = 10, 50, 100, 300$ respectively, where each plot corresponds a particular n. Because of the rectification function $|\langle \mathbf{I}, B_{x_i,s_i,\alpha_i} \rangle|$ in model (8), the model can be highly multi-modal. It seems that HMC can be easily trapped in the local modes, whereas the Gibbs sampler is less prone to the trapping of local modes. It is highly desirable to have a MCMC scheme that can traverse the local modes easily.

6.3. Local normalization and multiple selection

For the sake of recognition and detection accuracy, it is often desirable to perform some transformations of filter responses $\langle \mathbf{I}, B_{x,s,\alpha} \rangle$ on the observed images, so that the marginal distributions of the filter responses are closer to those produced by the Gaussian white noise model $q(\mathbf{I})$, which is used as the reference distribution or the background model. One type of transformation is local normalization:

$$\langle \mathbf{I}, B_{x,s,\alpha} \rangle \rightarrow \frac{\langle \mathbf{I}, B_{x,s,\alpha} \rangle}{\left(\sum_{x' \in \mathcal{N}_s(x)} \sum_{\alpha} \langle \mathbf{I}, B_{x',s,\alpha} \rangle^2 / N \right)^{1/2}}, \tag{20}$$

where $\mathcal{N}_s(x)$ is a local squared window centered at x. The size of $\mathcal{N}_s(x)$ is proportional to the scale s. N is the total number of wavelets whose centers are within $\mathcal{N}_s(x)$ and whose scales are s. The local normalization amounts to whitening the original images, and it tends to enhance the high frequency content of the original images. We can learn the model by matching the statistical properties of the synthesized images (without local normalization) to those of the observed images with local normalization. The

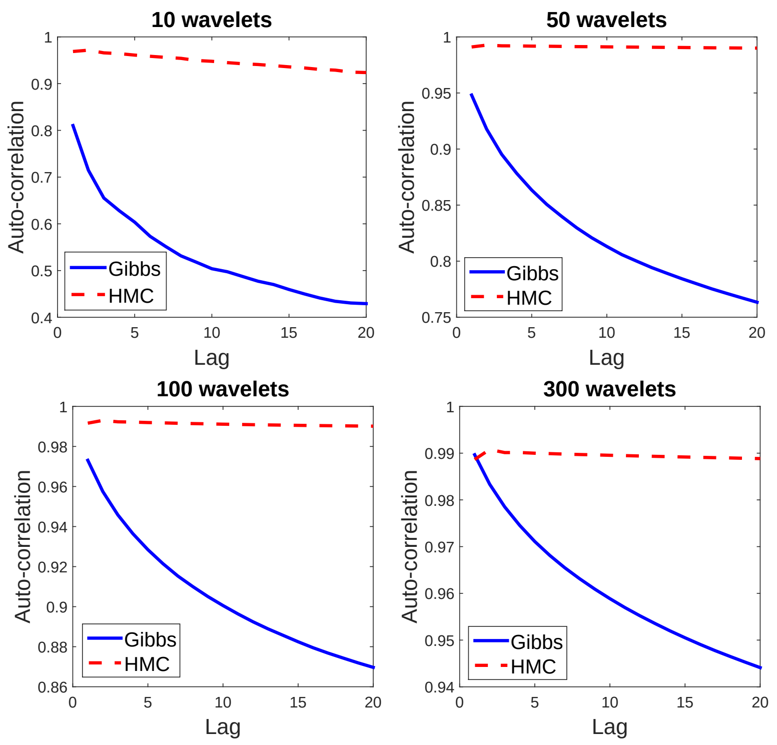


Figure 7: Comparison of auto-correlation between the Gibbs sampling method and the HMC sampling method. The number of selected wavelets varies from $n = 10, 50, 100, 300$.

synthesized images tend to have more prominent high frequency content, similar to the engraving style. The learned model is thus a model of the whitened versions of the training images. In our current implementation, the size of $\mathcal{N}_s(x)$ is about twice the size of the support of the filter $B_{x,s,\alpha}$.

To speed up the generative boosting algorithm, we may select multiple wavelets in each step. Specifically, we may divide the rectangular image domain into $k \times k$ cells of equal sizes. Then in each step, we can select one wavelet within each cell by maximizing $D(x, s, \alpha)$ in equation (10) over (x, s, α) within each cell, provided that the maximum in the cell is above a pre-specified threshold. This enables us to select up to k^2 wavelets within each iteration, thus accelerating the learning algorithm at the cost of the precision or optimality of the selection.

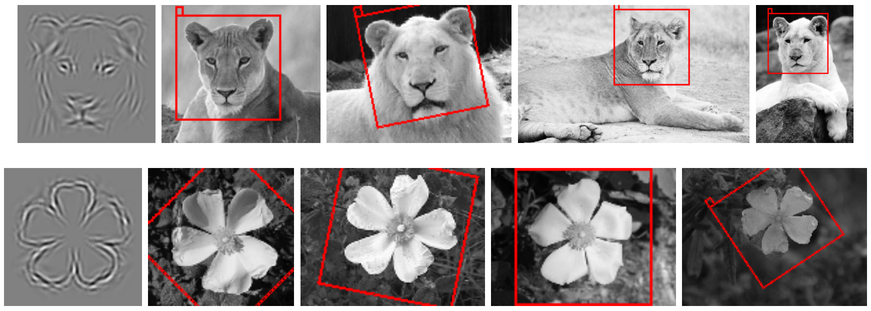


Figure 8: Learning from non-aligned images. Objects appear at different locations, scales and orientations in the training images. In each of the 2 experiments, the first image is the synthesized image generated from the learned model (local normalization of filter responses of the observed images is used, so the synthesized images have more prominent high frequency content). The rest of the images are some examples of training images and the bounding boxes indicate the inferred locations, scales and orientations of the objects. Lion experiment: each model has 500 wavelets, learned from 12 images. Flower experiment: each model has 500 wavelets, learned from 9 images. The sizes of templates are 100×100.

6.4. Learning from non-aligned images

The model can be learned from non-aligned images by iterating the following two steps: (1) detecting objects in the training images based on the current model. (2) re-learning the model from the detected image patches.

Figure 8 shows two experiments of learning from non-aligned images. We initialize the algorithm by randomly assigning an initial bounding box to each training image. The template size is 100×100. In detection, we search over 8 different resolutions of the images and 11 different orientations. The detected image patches are cropped from the optimal resolution for re-learning. The algorithm is run for 8 iterations. Each model has 500 wavelets. We use multiple selection in this experiment with 5×5 cells.

6.5. Learning mixture models for clustering tasks

We evaluate our learning algorithm on clustering tasks by fitting mixture models of sparse random field models using an EM-like algorithm. The algorithm iterates the following two steps: (1) classifying images into different clusters based on the current models of the clusters, (2) re-learning the model of each cluster from images

Table 1: Comparison of conditional purity and conditional entropy among sparse random field model learned by generative boosting, active basis model, two-step EM, and k-means with HOG on 12 clustering tasks. (Numbers in bold font indicate the best performances.)

(a) Conditional purity

Tasks (♯clusters)	ours	active basis	two-step EM	k-means + HOG
bull/cow (2)	**0.8867 ± 0.1981**	0.6667 ± 0.1269	0.8733 ± 0.0596	0.7600 ± 0.1690
cup/teapot (2)	**0.9067 ± 0.0983**	0.7867 ± 0.1346	0.8200 ± 0.1325	0.6400 ± 0.0925
plane/helicopter (2)	**0.9733 ± 0.0365**	0.9600 ± 0.0365	0.7133 ± 0.1386	0.7933 ± 0.1722
camel/elephant/deer (3)	**0.9200 ± 0.1419**	0.7289 ± 0.1519	0.7202 ± 0.1183	0.8000 ± 0.0861
clocks (3)	**0.9822 ± 0.0099**	0.6578 ± 0.0810	0.8578 ± 0.1375	0.8400 ± 0.1337
seagull/swan/eagle (3)	**1.0000 ± 0.0000**	0.8355 ± 0.1504	0.8000 ± 0.1728	0.9333 ± 0.1491
eye/mouth/ear/nose (4)	**0.8500 ± 0.1369**	0.8300 ± 0.1244	0.7734 ± 0.1146	0.8067 ± 0.1090
flowers (4)	**0.9200 ± 0.0960**	0.9033 ± 0.1120	0.7300 ± 0.0570	0.7800 ± 0.0681
PC components (4)	**0.9533 ± 0.1043**	0.9233 ± 0.1084	0.8500 ± 0.1296	0.8400 ± 0.1234
animal faces (5)	**0.8827 ± 0.0861**	0.7973 ± 0.0808	0.8693 ± 0.1013	0.7147 ± 0.1474
musical instruments (5)	**0.9227 ± 0.1060**	0.8880 ± 0.1134	0.7573 ± 0.1048	0.7840 ± 0.0614
animal bodies (5)	**0.8800 ± 0.0869**	0.8053 ± 0.1057	0.8133 ± 0.1015	0.7680 ± 0.0746
Average	**0.9231 ± 0.0917**	0.8152 ± 0.1105	0.7981 ± 0.1140	0.7883 ± 0.1155

(b) Conditional entropy

Tasks (♯clusters)	ours	active basis	two-step EM	k-means + HOG
bull/cow (2)	**0.2130 ± 0.2726**	0.5846 ± 0.1368	0.3451 ± 0.1273	0.4786 ± 0.1903
cup/teapot (2)	**0.2457 ± 0.1850**	0.4534 ± 0.2267	0.4040 ± 0.1823	0.6355 ± 0.0791
plane/helicopter (2)	**0.0821 ± 0.1124**	0.1390 ± 0.1152	0.5296 ± 0.1383	0.4337 ± 0.1714
camel/elephant/deer (3)	**0.1773 ± 0.2596**	0.5941 ± 0.2816	0.5935 ± 0.1625	0.4914 ± 0.1265
clocks (3)	**0.0665 ± 0.0372**	0.6581 ± 0.0770	0.3022 ± 0.2105	0.3327 ± 0.1791
seagull/swan/eagle (3)	**0.0000 ± 0.0000**	0.2595 ± 0.2373	0.3548 ± 0.2886	0.0924 ± 0.2067
eye/mouth/ear/nose (4)	**0.2080 ± 0.1898**	0.3211 ± 0.1390	0.4208 ± 0.2267	0.2724 ± 0.1527
flowers (4)	**0.1625 ± 0.1391**	0.1758 ± 0.1561	0.5521 ± 0.0564	0.5189 ± 0.1129
PC components (4)	**0.0669 ± 0.1497**	0.1693 ± 0.1881	0.2804 ± 0.2314	0.2649 ± 0.1586
animal faces (5)	**0.2859 ± 0.1608**	0.4472 ± 0.1655	0.3013 ± 0.1656	0.5155 ± 0.2156
musical instruments (5)	**0.1121 ± 0.1535**	0.2249 ± 0.1960	0.4861 ± 0.1313	0.3866 ± 0.1135
animal bodies (5)	**0.2902 ± 0.1517**	0.3537 ± 0.1414	0.4586 ± 0.1670	0.4772 ± 0.1372
Average	**0.1592 ± 0.1510**	0.3651 ± 0.1717	0.4190 ± 0.1740	0.4083 ± 0.1536

classified into this cluster. We also allow the objects to appear at unknown locations and orientations in the images. Therefore, this is an unsupervised learning task.

We compare our method with (a) the active basis model [23], (b) two-step EM [1], (c) k-means with HoG features [4].

We collect a dataset that consists of 12 clustering tasks. In each task, each cluster includes 15 images. The numbers of clusters vary from 2 to 5 and are assumed known in these tasks. The image ground-truth category labels are provided and assumed unknown to the algorithm, and they are used in evaluating the clustering accuracies. To quantize the clustering accuracies, we use two metrics: conditional purity and conditional entropy [21]. Given the true category label x and the inferred category label y of an image, the conditional purity is defined as $\sum_y p(y) \max_x p(x|y)$, and the conditional entropy is $\sum_y p(y) \sum_x p(x|y) \log(1/p(x/y))$, where both $p(y)$ and $p(x|y)$ are estimated from the training data. A better clustering algorithm would expect higher purity and lower entropy.

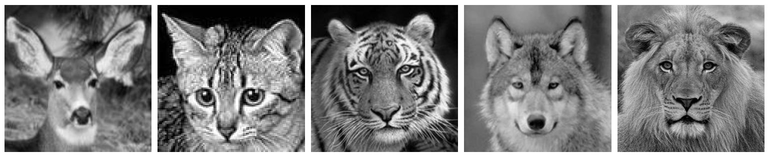
We fit a mixture of sparse models. $\tilde{M} = 100$ chains of sampled images are generated to estimate the parameters and normalizing constants. Typical template sizes are 100×100. The typical number of wavelets for each template is 600 or 700. We use multiple selection in this experiment with 5×5 cells. The dictionary consists of 3 scales of Gabor wavelets but no DoG wavelets in this experiment.

Table 1 shows the average clustering accuracies and standard errors based on 5 repetitions for 12 clustering tasks. The results show that our method performs better than other methods. Figure 9 illustrates one task of clustering animal heads. It displays one example image in each of the 5 categories. It also displays synthesized images generated by the learned sparse models. We use local normalization in this experiment, so the synthesized images emphasize high frequency content.

6.6. Comparison with two-stage learning

A two-stage learning algorithm was proposed in our previous work [24] to train the sparse model by taking advantage of the shared sparse coding model (14).

(1) In the first stage, a simultaneous or shared sparse coding scheme is used to select $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ by simultaneously reconstructing all the observed


(a) observed images

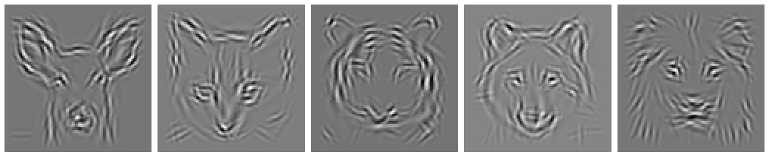

(b) synthesized images

Figure 9: Unsupervised learning of mixture of sparse models. (a) displays one training image for each cluster. (b) displays one synthesized image generated by the model learned for each cluster (local normalization of filter responses of the observed images is used, so the synthesized images have more prominent high frequency content). The sizes of the images are 100 $\times$ 100. The number of images within each cluster is 15. The number of clusters is 5. The number of selected wavelets is 700.

images. The selection is accomplished by minimizing the least squares reconstruction error for all the training images

$$\sum_{m=1}^{M} \|\mathbf{I}_m - \sum_{i=1}^{n} c_{m,i} B_{x_i+\Delta x_{m,i}, s_i, \alpha_i+\Delta\alpha_{m,i}}\|^2. \tag{21}$$

The minimization of (21) can be accomplished by a shared matching pursuit algorithm, which sequentially adds wavelets to reduce the above reconstruction error. Figure 3 in Section 2.1 illustrates the basic idea. It corresponds to a boosting algorithm whose objective function is (21) instead of the log-likelihood.

(2) After selecting $\mathbf{B} = (B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$, the second stage estimates λ_i by maximum likelihood using the stochastic gradient ascent algorithm as in equation (5).

To compare the generative boosting algorithm with the above two-stage algorithm based on shared matching pursuit, we conduct two experiments that compare the two methods using two criteria.

Approximation error. The sparse model (8) is a sparsified version of the dense model (1). The quantity $D(x, s, \alpha)$ in equation (10) measures the discrepancy between the fitted model and the observed data as far as the wavelet $B_{x,s,\alpha}$ is concerned. We can compute the average of $|D(x, s, \alpha)|$ over all x, s, α as an overall measure of dis-

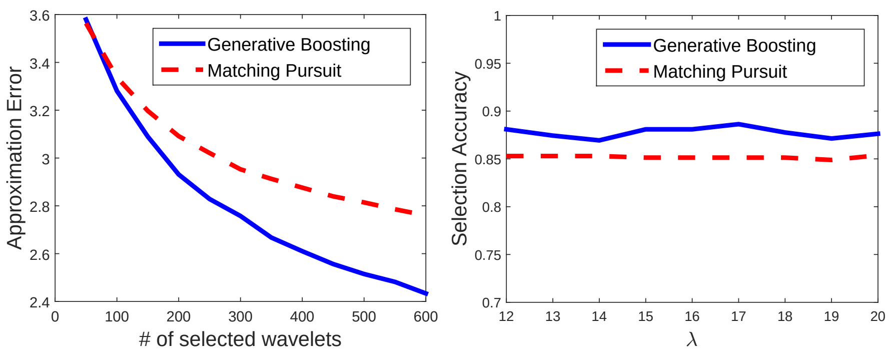


Figure 10: Comparison between our method (generative boosting) and the two-stage learning method (shared matching pursuit). *Left*: Approximation error measured by the average discrepancy between the learned model and the observed examples in terms of expectations of rectified filter responses. *Right*: Selection accuracy measured by the correlation between the true wavelets in the pre-defined model and the selected wavelets by the two methods.

crepancy or approximation error. The left panel of Figure 10 plots the approximation error versus the number of wavelets selected for the generative boosting method and the shared matching pursuit method respectively for learning from the cat images. It can be seen that generative boosting leads to a faster reduction of the approximation error.

Selection accuracy. In this experiment, we use a pre-defined model with known $\mathbf{B} = (B_i = B_{x_i,s_i,\alpha_i}, i = 1, ..., n)$ and $\lambda = (\lambda_i, i = 1, ..., n)$ as the true model to generate training images so that we know the ground truth. The pre-defined model has the shape of a square, and we allow some overlap between the wavelets B_i that make up the shape of the square. We use $n = 16$ wavelets in this experiment, and we generate $M = 36$ training images, assuming that all λ_i are equal for simplicity. Let $\hat{\mathbf{B}} = (\hat{B}_i, i = 1, ..., n)$ be the selected wavelets learned from the training images, where n is assumed known. We use the correlation between $\sum_{i=1}^{n} B_i$ and $\sum_{i=1}^{n} \hat{B}_i$ as the measure of accuracy of recovering the true wavelets. The right panel of Figure 10 plots the selection accuracy versus typical values of λ_i for generative boosting and shared matching pursuit. The accuracy is computed by averaging over 10 repetitions. It can been seen that generative boosting is consistently more accurate than shared matching pursuit. In this experiment, we use the Gibbs sampling of filter responses, and the wavelet selection is based on the noiseless synthesized images. Before selecting each new wavelet, we estimate the parameters of the current model by maximum likelihood.

7. Conclusion

Much success has been gained by researchers in harmonic analysis and other fields by exploiting generic sparsity via regularization. Yet little progress has been made on representing and learning specific sparsity patterns. The sparse random field model studied in this paper is a candidate for such a representation. This paper provides a computational core for sampling from and learning such a model.

The learned models can be mapped to the second-layer nodes of CNN that are sparsely connected to the first layer nodes of Gabor-like filters. Both the sparse connections and the generative nature of the learned models make them more explicit and meaningful than common CNN nodes.

The mixture model experiments show that the models can be learned in an unsupervised fashion. It may be possible to extend the model to a multi-layer convolutional structure with sparse connections learned by generative boosting. Such a multi-layer model may lead to high-level representations of sparse-land that go beyond linear basis functions or wavelets.

Reproducibility

`http://www.stat.ucla.edu/~jxie/GenerativeBoosting.html` The page contains the datasets and matlab/C code for producing the experimental results presented in this paper.

Acknowledgements

We thank the two anonymous reviewers for their insightful comments and valuable suggestions that have helped us improve the content and presentation of this paper. The work is supported by NSF DMS 1310391, ONR MURI N00014-10-1-0933, DARPA MSEE FA8650-11-1-7149.

Appendix

Technical details on wavelet sparse coding

For notational simplicity, we use B_i to denote B_{x_i,s_i,α_i}. For the sparse model, the number of selected wavelets, $n \ll |\mathcal{D}|$, the number of pixels in image domain $\mathcal{D}$. Let $\mathbf{I}$ and B_i be $|\mathcal{D}| \times 1$ vectors. For the $|\mathcal{D}| \times n$ matrix $\mathbf{B} = (B_1, ..., B_n)$, we can construct a $|\mathcal{D}| \times (|\mathcal{D}| - n)$ complementary matrix $\bar{\mathbf{B}} = (\bar{B}_{n+1}, ..., \bar{B}_{|\mathcal{D}|})$, so that $\bar{\mathbf{B}}^\top \bar{\mathbf{B}} = I_{|\mathcal{D}|-n}$, the $|\mathcal{D}| - n$ dimensional identity matrix, and $\bar{\mathbf{B}}^\top \mathbf{B} = 0$, the zero matrix. That is, the column vectors in $\bar{\mathbf{B}}$ are orthonormal, and they are orthogonal to the column vectors in $\mathbf{B}$. We shall make $\bar{\mathbf{B}}$ implicit in the end.

Now let $\tilde{\mathbf{B}} = (\mathbf{B}, \bar{\mathbf{B}})$ be the $|\mathcal{D}| \times |\mathcal{D}|$ squared matrix. Then we can write

$$\mathbf{I} = \sum_{i=1}^{n} c_i B_i + \sum_{i=n+1}^{|\mathcal{D}|} \bar{c}_i \bar{B}_i = \mathbf{B}C + \bar{\mathbf{B}}\bar{C} = \tilde{\mathbf{B}}\tilde{C}, \tag{22}$$

where $C = (c_i, i = 1, ..., n)$ is the $n \times 1$ vector, and $\bar{C} = (\bar{c}_i, i = n+1, ..., |\mathcal{D}|)$ is the $(|\mathcal{D}| - n) \times 1$ vector, and $\tilde{C} = (C, \bar{C})$ is the $|\mathcal{D}| \times 1$ vector. $\tilde{C} = \tilde{\mathbf{B}}^{-1}\mathbf{I}$. C is the least squares regression coefficients of $\mathbf{I}$ on $\mathbf{B}$, i.e., $C = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{I} = (\mathbf{B}^\top\mathbf{B})^{-1}R$, where $R = \mathbf{B}^\top\mathbf{I}$, i.e., $R = (r_i, i = 1, ..., n)$ and $r_i = \langle \mathbf{I}, B_i \rangle$ is the filter response. $\bar{C} = \bar{\mathbf{B}}^\top\mathbf{I}$. Let $\epsilon = \bar{\mathbf{B}}\bar{C}$, we can write $\mathbf{I} = \sum_{i=1}^{n} c_i B_i + \epsilon$, where ϵ is the residual image after we project $\mathbf{I}$ onto the subspace spanned by $\mathbf{B}$.

The distribution $p(\mathbf{I}; \mathbf{B}, \lambda)$ in (8) induces a joint distribution $p_{\tilde{C}}(\tilde{C})$ via the transformation $\mathbf{I} = \tilde{\mathbf{B}}\tilde{C}$, or $\tilde{C} = \tilde{\mathbf{B}}^{-1}\mathbf{I}$. We shall show that under $p_{\tilde{C}}(\tilde{C})$, $C \sim p_C(C; \lambda)$ and $\bar{C} \sim p_{\bar{C}}(\bar{C})$ are independent of each other. In fact, $p_{\bar{C}}(\bar{C})$ is Gaussian white noise, and $\epsilon = \bar{\mathbf{B}}\bar{C}$ is Gaussian white noise projected onto $\bar{\mathbf{B}}$.

Specifically, we can write the original model (8) in matrix notation

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |\mathbf{B}^\top\mathbf{I}| \rangle\right) q(\mathbf{I}), \tag{23}$$

where for a vector v, $|v|$ is a vector of the same dimension where we take absolute values element-wise. The distribution of $\tilde{C}$ can then be derived via

$$p(\mathbf{I}; \lambda) d\mathbf{I} = p(\tilde{\mathbf{B}}\tilde{C}; \lambda) d\tilde{\mathbf{B}}\tilde{C} = p(\tilde{\mathbf{B}}\tilde{C}; \lambda)|\tilde{\mathbf{B}}| d\tilde{C} = p_{\tilde{C}}(\tilde{C}; \lambda) d\tilde{C}, \tag{24}$$

where $|\tilde{\mathbf{B}}|$ denotes the absolute value of the determinant of $\tilde{\mathbf{B}}$, the Jacobian term. $|\tilde{\mathbf{B}}|^2 = |\tilde{\mathbf{B}}^\top \tilde{\mathbf{B}}| = |\mathbf{B}^\top \mathbf{B}|$, so $|\tilde{\mathbf{B}}| = |\mathbf{B}^\top \mathbf{B}|^{1/2}$. Thus,

$$
\begin{aligned}
& p_{\tilde{C}}(\tilde{C};\lambda) = p(\tilde{\mathbf{B}}\tilde{C};\lambda)|\tilde{\mathbf{B}}| \\
& = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |\mathbf{B}^\top(\mathbf{B}C + \bar{\mathbf{B}}\bar{C})|\rangle\right) q(\tilde{\mathbf{B}}\tilde{C})|\tilde{\mathbf{B}}| \\
& = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |\mathbf{B}^\top \mathbf{B}C|\rangle\right) \frac{1}{(2\pi\sigma^2)^{|\mathcal{D}|/2}} \exp\left(-\frac{1}{2\sigma^2}||\tilde{\mathbf{B}}\tilde{C}||^2\right)|\tilde{\mathbf{B}}| \\
& = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |\mathbf{B}^\top \mathbf{B}C|\rangle\right) \frac{1}{(2\pi\sigma^2)^{|\mathcal{D}|/2}} \exp\left(-\frac{1}{2\sigma^2}(C^\top(\mathbf{B}^\top\mathbf{B})C + \|\bar{C}\|^2)\right)|\mathbf{B}^\top\mathbf{B}|^{1/2} \\
& = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |\mathbf{B}^\top \mathbf{B}C|\rangle\right) \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(C^\top(\mathbf{B}^\top\mathbf{B})C)\right)|\mathbf{B}^\top\mathbf{B}|^{1/2} \\
& \times \frac{1}{(2\pi\sigma^2)^{(|\mathcal{D}|-n)/2}} \exp\left(-\frac{1}{2\sigma^2}\|\bar{C}\|^2\right) = p_C(C;\lambda)p_{\bar{C}}(\bar{C}),
\end{aligned} \tag{25}
$$

where C and $\bar{C}$ are independent of each other, $p_{\bar{C}}(\bar{C})$ is $|\mathcal{D}| - n$ dimensional Gaussian white noise, i.e., $\bar{c}_i \sim \mathrm{N}(0, \sigma^2)$ independently for $i = n+1, ..., |\mathcal{D}|$. The distribution of the coefficients

$$
p_C(C;\lambda) = \frac{1}{Z(\lambda)} \exp\left(\langle \lambda, |\mathbf{B}^\top \mathbf{B}C|\rangle\right) q_C(C), \tag{26}
$$

where

$$
q_C(C) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(C^\top(\mathbf{B}^\top\mathbf{B})C)\right)|\mathbf{B}^\top\mathbf{B}|^{1/2} \tag{27}
$$

is a multivariate Gaussian distribution, $C \sim \mathrm{N}(0, \sigma^2(\mathbf{B}^\top\mathbf{B})^{-1})$. Thus we can write the model in the generative form $C \sim p_C(C;\lambda)$, $\mathbf{I} = \mathbf{B}^\top C + \epsilon$, where $\epsilon = \bar{\mathbf{B}}\bar{C}$, and is independent of C.

545 *Technical details on moving along basis vectors*

For Gibbs sampling of $C \sim p_C(C;\lambda)$, we can update the i-th component by first sampling from $p(d_i) \propto p(\mathbf{I} + d_i B_i;\lambda)$, and letting $c_i \leftarrow c_i + d_i$, $\mathbf{I} \leftarrow \mathbf{I} + d_i B_i$, where d_i is a scalar increment of c_i. To see this, let us drop the subscript i for simplicity, and write $p(\mathbf{I};\lambda)$ in (23) simply as $p(\mathbf{I})$. We shall explain why $p(d) \propto p(\mathbf{I} + dB)$.

For the basis vector B, let $\bar{\mathbf{B}}$ be the $|\mathcal{D}| \times (|\mathcal{D}| - 1)$ complementary matrix, whose columns are linearly independent of each other and independent of B (here we use the notation $\bar{\mathbf{B}}$ in a slightly different manner from the above; this new $\bar{\mathbf{B}}$ corresponds to $(B_1, ..., B_{i-1}, B_{i+1}, ..., B_n, \bar{\mathbf{B}})$ defined above). Let $\tilde{\mathbf{B}} = (B, \bar{\mathbf{B}})$, and $\tilde{C} = (c, \bar{C})$ (again, here we use the notation $\tilde{C}$ and $\bar{C}$ in a slightly different manner from the above. Here $\bar{C}$ corresponds to $(c_1, ..., c_{i-1}, c_{i+1}, ...c_n, \bar{C})$ defined above). So $\mathbf{I} = \tilde{\mathbf{B}}\tilde{C} = cB + \bar{\mathbf{B}}\bar{C}$, where $\tilde{C} = \tilde{\mathbf{B}}^{-1}\mathbf{I}$. The joint distribution of $\tilde{C} = (c, \bar{C})$ is $p_{\tilde{C}}(c, \bar{C}) = p(cB + \bar{\mathbf{B}}\bar{C})|\tilde{\mathbf{B}}|$. Therefore, the conditional distribution (as a function of c)

$$\begin{aligned} p_{\tilde{C}}(c|\bar{C}) &= \frac{p_{\tilde{C}}(c, \bar{C})}{\int_c p_{\tilde{C}}(c, \bar{C})dc} \\ &= \frac{p(cB + \bar{\mathbf{B}}\bar{C})}{\int_c p(cB + \bar{\mathbf{B}}\bar{C})dc} \propto p(cB + \bar{\mathbf{B}}\bar{C}), \end{aligned} \tag{28}$$

where the Jacobian term $|\tilde{\mathbf{B}}|$ is canceled. Let c_0 be the current value of c. Let $\mathbf{I}_0 = c_0B + \bar{C}\bar{\mathbf{B}}$ be the current image. In the Gibbs sampler, given the current value of $\bar{C}$, we need to draw a new c from $p_{\tilde{C}}(c|\bar{C})$ to replace the current c_0. Then the conditional distribution of this new c (where c_0 and $\bar{C}$ are given) is

$$\begin{aligned} p_{\tilde{C}}(c|\bar{C}) \propto p(cB + \bar{\mathbf{B}}\bar{C}) &= p(c_0B + \bar{\mathbf{B}}\bar{C} + (c - c_0)B) \\ &= p(\mathbf{I}_0 + (c - c_0)B). \end{aligned} \tag{29}$$

Let $d = c - c_0$. Then the conditional distribution of d given $\bar{C}$ is $p(d|\bar{C}) \propto p(\mathbf{I}_0 + dB)$, where the Jacobian of the translation $d = c - c_0$ is 1.

Technical details on Gibbs sampling of filter responses

For the positive piece with $r_i \geq 0$, the mean of the Gaussian is $\mu_+ = (-\sum_{j \neq i} a_{ij}r_j + \sigma^2\lambda)/a_{ii}$. The variance of the Gaussian is $s^2 = \sigma^2/a_{ii}$. For the negative piece with $r_i < 0$, the mean of the Gaussian is $\mu_- = (-\sum_{j \neq i} a_{ij}r_j - \sigma^2\lambda)/a_{ii}$. The variance of the Gaussian is again $s^2 = \sigma^2/a_{ii}$. Let $\Phi(x; \mu, \sigma^2)$ be the cumulative density function of the normal distribution $\mathrm{N}(\mu, \sigma^2)$. Then the area under the Gaussian density curve of $\mathrm{N}(\mu_+, s^2)$ restricted to the positive part is $\rho_+ = 1 - \Phi(0; \mu_+, s^2)$. The area under the Gaussian density curve of $\mathrm{N}(\mu_-, s^2)$ restricted to the negative part is $\rho_- = \Phi(0; \mu_-, s^2)$. For the piecewise Gaussian distribution $p_R(r_i|r_{-i})$, the probability $r_i \geq 0$ is $p_+ \propto \rho_+ \exp(\mu_+^2/2s^2)$, and the probability $r_i < 0$ is $p_- \propto \rho_- \exp(\mu_-^2/2s^2)$, with $p_+ + p_- = 1$. In order to sample from $p_R(r_i|r_{-i})$, we can first decide whether r_i is positive or negative according to p_+ and p_-, and then sample from the corresponding truncated normal distribution (using the inversion method).

References

[1] Adrian Barbu, Tianfu Wu, and Ying Nian Wu. Learning mixtures of bernoulli templates by two-round em with performance guarantee. *Electronic Journal of Statistics*, 8(2):3004–3030, 2014.

[2] Jie Chen and Xiaoming Huo. Sparse representations for multiple measurement vectors (mmv) in an over-complete dictionary. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 4, pages 257–260. IEEE, 2005.

[3] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[5] Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393, 1997.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[7] Michael Elad. *Sparse and Redundant Representations: from Theory to Applications in Signal and Image Processing*. Springer Science & Business Media, 2010.

[8] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and System Sciences*, 55(1):119–139, 1997.

[9] Jerome H Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.

[10] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[11] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6(6):721–741, 1984.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Julien Mairal, Francis Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8:85–283, 2014.

[15] Radford M Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[16] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[17] Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *The Journal of Machine Learning Research*, 5:941–973, 2004.

[18] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[19] Joel A Tropp, Anna C Gilbert, and Martin J Strauss. Algorithms for simultaneous sparse approximation. part i: greedy pursuit. *Signal Processing - Sparse Approximations in Signal and Image Processing*, 86(3):572–588, 2006.

[20] Zhuowen Tu. Learning generative models via discriminative approaches. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[21] Tinne Tuytelaars, Christoph H Lampert, Matthew B Blaschko, and Wray Buntine. Unsupervised object discovery: A comparison. *International Journal of Computer Vision*, 88(2):284–302, 2010.

[22] Max Welling, Richard S Zemel, and Geoffrey E Hinton. Self supervised boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 665–672, 2002.

[23] Ying Nian Wu, Zhangzhang Si, Haifeng Gong, and Song-Chun Zhu. Learning active basis model for object detection and recognitio. *International Journal of Computer Vision*, 90:198–235, 2010.

[24] Jianwen Xie, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Learning sparse frame models for natural image patterns. *International Journal of Computer Vision*, pages 1–22, 2014.

[25] Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4):177–228, 1999.

[26] Song Chun Zhu, Xiuwen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient markov chain monte carlo - towards a 'trichromacy' theory of texture. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22:245–261, 2000.

[27] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.