Annals of Mathematical Sciences and Applications
Volume 3, Number 1, 211–254, 2018

Sparse and deep generalizations of the FRAME model

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

In the pattern theoretical framework developed by Grenander and advocated by Mumford for computer vision and pattern recognition, different patterns are represented by statistical generative models. The FRAME (Filters, Random fields, And Maximum Entropy) model is such a generative model for texture patterns. It is a Markov random field model (or a Gibbs distribution, or an energy-based model) of stationary spatial processes. The log probability density function of the model (or the energy function of the Gibbs distribution) is the sum of translation-invariant potential functions that are one-dimensional non-linear transformations of linear filter responses. In this paper, we review two generalizations of this model. One is a sparse FRAME model for non-stationary patterns such as objects, where the potential functions are location specific, and they are non-zero only at a selected collection of locations. The other generalization is a deep FRAME model where the filters are defined by a convolutional neural network (CNN or ConvNet). This leads to a deep convolutional energy-based model. The local modes of the energy function satisfies an auto-encoder which we call the Hopfield auto-encoder. The model can be learned by an "analysis by synthesis" algorithm that iterates a sampling step for synthesis and a learning step for analysis. The algorithm admits an adversarial interpretation where the learning step and sampling step play a minimax game based on a value function. We can recruit a generator model as a direct and approximate sampler of the deep energy-based model to speed up the sampling step, and the two models can be learned simultaneously by a cooperative learning algorithm.

Keywords and phrases: Adversarial interpretation, convolutional neural network, cooperative learning, energy-based model, generator model, Hopefield auto-encoder, sparse coding.

1. Introduction

1.1. Pattern theory and generative models

Pattern theory is a theoretical and computational framework developed by Grenander [33] and advocated by Mumford [62] for computer vision and

pattern recognition. In this framework, the patterns are represented by statistical generative models that are in the form of probability distributions of the signals such as images. Intuitively, such models tell us what the patterns look like, e.g., what a cat looks like and what a dog looks like. The models can be learned from the observed training examples, e.g., images of cats, or images of dogs, often via an "analysis by synthesis" scheme, where the parameters of the models are updated to make the synthesized examples generated by the learned models to be similar to the observed examples, e.g., a model of cats can generate images that are similar to the observed images of cats. With the learned models of the patterns, pattern recognition can be accomplished by likelihood-based or Bayesian inference, e.g., the knowledge of what cats and dogs look like enables us to recognize cats and dogs from testing images by matching the images to the models.

More specifically, the generative models can be useful in the following scenarios of learning. (1) Unsupervised learning, where the observed data are not annotated or labeled, for instance, we observe images of cats and dogs, but do not know which images are cats and which images are dogs. The generative models enable us to learn features or hidden structures (such as clusters or attributes) in the data. (2) Semi-supervised learning, where only part of the data are annotated. The generative models enable us to make better use of the unlabeled data for the purpose of classification or prediction. (3) Supervised learning from small data. The generative models enable us to learn from the data more efficiently (in terms of statistical accuracy). (4) Reinforcement learning in the Markov decision process framework. The generative models can be used for model-based inference of the states and model-based planning of the action policies. In addition to learning, realistic generative models are also useful for computer graphics.

1.2. FRAME model of texture patterns

In this paper, we shall review a particular generative model called FRAME (Filters, Random field, And Maximum Entropy) model developed by Zhu, Wu, and Mumford [99, 86, 97] and its recent generalizations.

The FRAME model was originally developed for modeling texture patterns, which are ubiquitous in natural scenes, such as grasses, tree leaves, brick walls, water waves, etc. The problem of texture perception in preattentive vision was extensively studied by Julesz, whose pursuit for a solution to this problem was driven by the following two fundamental questions. (1) What are the statistical properties that define a texture? [46] (2) What are the basic elements, or textons, that constitute a texture? [47] For the first

question, researchers have studied second order statistics, k-gon statistics, etc [96]. For instance, mathematicians Diaconis and Freedman [14] designed image pairs with the same second order statistics but different texture patterns. The second question inspired Marr [59] to propose the theory of primal sketch based on image primitives or local image tokens detected by some local image operators or feature extractors. An important advance was made by Heeger and Bergen [36], who showed that realistic texture patterns can be synthesized by matching the marginal histograms of responses from a bank of linear filters.

Inspired by the idea that texture statistics can be defined by the marginal histograms of filter responses, Zhu, Wu, and Mumford [99] developed the FRAME model as the maximum entropy model that is constrained by such statistics. The resulting maximum entropy model is a Markov random field model or an energy-based model [53] in the form of a Gibbs distribution [2, 28]. Originated from statistical physics, the Markov random field models or the Gibbs distributions are an important class of probability models for spatial processes such as those observed in natural images. The log probability density function of a Markov random field model or the energy function of the Gibbs distribution is the sum of potential functions defined on the so-called cliques that consist of neighboring sites or pixels. The potential functions can be high-dimensional for those cliques that consist of many pixels, and it is difficult to learn such high-dimensional potential functions from the data. The FRAME model solves this problem by recruiting a bank of linear filters, and parametrizing the potential functions as one-dimensional non-linear transformations of linear filter responses. This model is the maximum entropy distribution that reproduces the marginal statistics such as marginal histograms of the filter responses, where for each filter, the marginal histogram is pooled over all the pixels in the image domain. The bank of filters can be designed, such as Gabor filters or Gabor wavelets [9] tuned to different locations, scales and orientations. They can also be learned, together with the non-linear transformations, from the training data.

Another justification for the FRAME model is via the so-called Julesz ensemble [86, 95], which is the uniform distribution over the set of images with the same marginal histograms of filter responses. If the image size is large, then the probability distribution of local image patches is given by the FRAME model. The set of images constrained by certain feature statistics can be used to define a certain concept such as a texture pattern. It is related to the micro-canonical ensemble in statistical physics [29], which is the set of configurations with the same energy. Under the uniform distribution on the micro-canonical ensemble, the local system follows the Gibbs distribution or the canonical ensemble.

1.3. Sparse and deep generalizations

The FRAME model is originally developed for modeling spatially stationary processes such as stochastic textures, where the potential functions are translation invariant. In this article, we review two generalizations of the FRAME model. One is a sparse FRAME model [87, 89] where the potential functions are location specific, and they are non-zero only at selected locations. This model is intended to model image patterns that are non-stationary in the spatial domain, such as object patterns, e.g., images of cats and dogs. The model can be written as a shared sparse coding model, where the observed images are represented by a commonly shared set of wavelets selected from a dictionary. In this shared sparse coding model, the original linear filters for bottom-up computation (from image to filter responses) become linear basis functions for top-down representation (from coefficients to image).

The second generalization of the FRAME model is inspired by the recent successes of deep convolutional neural networks (CNNs or ConvNets) [52, 50], and it can be called the deep FRAME model [57]. The filters used in the original FRAME model are linear filters that capture local image features. In the deep FRAME model, the linear filters are replaced by the non-linear filters at a certain convolutional layer of a pre-trained deep ConvNet. Such filters can capture more complex patterns, and the deep FRAME model built on such filters can be more expressive.

Instead of using filters from a pre-trained ConvNet, we can also learn the filters from the observed data. The resulting model is a deep convolutional energy-based model [65, 8, 90] or what can be called a generative ConvNet model [90]. Such a model can be considered a recursive multi-layer generalization of the original FRAME model. The log probability density function of the original FRAME model consists of non-linear transformations of linear filter responses. If we repeat this structure recursively, we get the generative ConvNet model with multiple layers of linear filtering followed by point-wise non-linear transformations. It is possible to learn such a model from natural images [90].

We can generate synthetic images by sampling from the above FRAME models using Markov chain Monte Carlo (MCMC) such as the Langevin dynamics [55, 30], which runs gradient descent on the energy function of the model while adding Gaussian white noises for diffusion. This sampling scheme was first applied to the original FRAME model by Zhu and Mumford (1998) [97], where the gradient descent part of the dynamics was interpreted as the Gibbs Reaction And Diffusion Equation (GRADE). The Langevin dynamics can be used to sample from deep FRAME model where the gradient can be efficiently computed by back-propagation.

1.4. Auto-encoder, adversarial interpretation, generator as sampler

The FRAME model can be written as exponential tilting of a reference distribution such as the uniform measure or the Gaussian white noise model. If the reference distribution is the Gaussian white noise model, the local modes of the probability density follow an auto-encoder. We call it the Hopfield auto-encoder, because it defines the local energy minima of the model [43]. In the Hopfield auto-encoder, the bottom-up filters detect the patterns corresponding to the filters, then the binary detection results are used as the coefficients in the top-down representation where the original filters play the role of basis functions.

The learning of the FRAME model follows the analysis by synthesis scheme [33]. We can use the Langevin dynamics to sample from the current model to generate synthetic images. Then we update the model parameters based on the statistical difference between the observed images and the synthetic images, so that the model shifts its probability density function, especially the high density regions or the low energy regions, from the synthetic images to the observed images. In the zero temperature limit, this learning and sampling algorithm admits an adversarial interpretation, where the learning step and the sampling step play a minimax game based on a value function.

The sampling of the FRAME model requires iterative MCMC such as Langevin dynamics or Hamiltonian Monte Carlo [63]. A recently proposed generator model [32, 49, 71, 60] can be recruited as a much more efficient non-iterative sampler that replaces the MCMC sampling by direct ancestral sampling [88].

The rest of the paper is organized as follows. Section 2 introduces the original FRAME model. Section 3 presents the sparse FRAME model. Section 4 presents the deep FRAME model. Section 5 explains the Hopfield auto-encoder structure. Section 6 introduces the generator model as a non-iterative sampler. Section 7 presents the adversarial interpretation. Section 8 concludes with a discussion.

2. FRAME model

The original FRAME model [99, 86, 97] is based on linear filters. We shall first review the linear filters as well as linear basis functions. Then we present the FRAME model.

2.1. Filters and basis functions

Let $\mathbf{I}(x)$ be an image defined on the square (or rectangular) domain $\mathcal{D}$, where $x = (x_1, x_2)$ (a two-dimensional vector) indexes the coordinates of pixels. We can treat $\mathbf{I}(x)$ as a two-dimensional function defined on $\mathcal{D}$. We can also treat $\mathbf{I}$ as a vector if we fix an ordering for the pixels. Let $D = |\mathcal{D}|$ count the number of pixels in $\mathcal{D}$, i.e., D is the dimensionality of the vector $\mathbf{I}$.

A linear filter is a local weighted sum of image intensities around each pixel. A linear basis function (or basis vector) is a local image patch intended to represent image intensities.

Suppose we have a set of linear filters $\{F_k, k = 1, ..., K\}$. We can apply each F_k to image $\mathbf{I}$ to obtain a filtered image or feature map, denoted by $F_k * \mathbf{I}$, which is of the same size as $\mathbf{I}$ and is also defined on $\mathcal{D}$ (with proper handling of boundaries). Let $[F_k * \mathbf{I}](y)$ be the filter response or feature at position y, then

$$[F_k * \mathbf{I}](y) = \sum_{x \in \mathcal{S}} w_{k,x} \mathbf{I}(y + x), \tag{1}$$

where the weights $w_{k,x}$ define the filter F_k, and $\mathcal{S}$ is the localized support of the filter centered at the origin. See Figure 1 for an illustration, where $\mathcal{S}$ is 3×3, and $\mathcal{D}$ is 6×6. In practice, both $\mathcal{S}$ and $\mathcal{D}$ can be much larger. $\mathcal{S}$ can be different for different F_k. The filtering operation is also a convolution operation.

Suppose we have a set of prototype basis functions or wavelets $\{B_k(x), k = 1, ..., K\}$. We assume that each B_k is supported on a local domain $\mathcal{S}$ centered at the origin. Again, $\mathcal{S}$ may be different for different B_k. We can spatially shift or translate B_k to a position y to get a translated copy of B_k, denoted by $B_{k,y}(x) = B_k(x - y)$. We can treat each $B_{k,y}(x)$ as a function defined on $x \in \mathcal{D}$, just as $\mathbf{I}$, except that $B_{k,y}(x)$ is locally supported. We can also treat $B_{k,y}$ as a vector of the same dimensionality as $\mathbf{I}$.

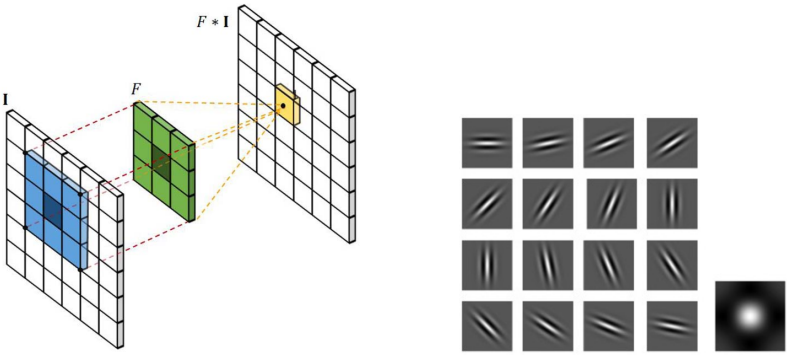
The basis functions are used in the linear representation

$$\mathbf{I}(x) = \sum_{k,y} c_{k,y} B_{k,y}(x) + \epsilon(x), \tag{2}$$

where $c_{k,y}$ are the coefficients, often assumed to be sparse, and $\epsilon(x)$ is the residual image.

The inner product between $\mathbf{I}$ and $B_{k,y}$ is

$$\langle \mathbf{I}, B_{k,y} \rangle = \sum_{x \in \mathcal{D}} \mathbf{I}(x) B_{k,y}(x) = \sum_{x \in \mathcal{S}} \mathbf{I}(y + x) B_k(x). \tag{3}$$



(a) (b)

Figure 1: (a) Filtering or convolution: applying a filter F (3×3) on an image $\mathbf{I}$ (6×6) to get a filtered image (6×6) or feature map $F * \mathbf{I}$. Each pixel of $F * \mathbf{I}$ is computed by the weighted sum of the 3×3 pixels of $\mathbf{I}$ centered at this pixel. (b) Gabor filters (wavelets) at different orientations, and Difference of Gaussians (DoG) filter (the rightmost one). The Gabor wavelets are sine and cosine waves multiplied by elongated Gaussian functions. The DoG wavelet is isotropic. The wavelets can appear at different locations and scales.

The connection between $B_{k,y}$ and F_k is

$$\langle \mathbf{I}, B_{k,y} \rangle = [F_k * \mathbf{I}](y) \tag{4}$$

if $B_k(x) = w_{k,x}$. Examples of basis functions or filters include oriented and elongated Gabor wavelets [9] as well as isotropic Difference of Gaussian (DoG) wavelets as illustrated by Figure 1. In subsequent sections, we shall often drop the argument x in $\mathbf{I}(x)$ and $B_{k,y}(x)$, and treat them as vectors.

While the filters are about bottom-up feature extraction (bottom-up means from the image to the filter responses), the basis functions are about top-down linear representation (top-down means from the coefficients to the image). It is desirable to unify these two roles in the same model.

2.2. Sparse representation

Assume we are given a dictionary of wavelets or basis functions $\{B_{k,x}\}$, where k may index a finite collection of prototype functions $\{B_k, k = 1, ..., K\}$, and where $B_{k,x}$ is a spatially translated copy of B_k to position x. We can

represent an image $\mathbf{I}$ by

$$\mathbf{I} = \sum_{k,x} c_{k,x} B_{k,x} + \epsilon, \tag{5}$$

where $c_{k,x}$ are the coefficients, and ϵ is the residual image. It is often assumed that the representation is sparse, i.e., most of the $c_{k,x}$ are equal to zero. The resulting representation is also called sparse coding [66, 20].

The sparsification of $c_{k,x}$, i.e., the selection of the basis functions, can be accomplished by matching pursuit [58] or basis pursuit/Lasso [5, 78]. Using a Lasso-like objective function, the dictionary of basis functions $\{B_k\}$ can be learned from a collection of training images [66, 19]. It is sometimes called sparse component analysis [17]. It can be considered a generalization of factor analysis. For natural images, the basis functions learned resemble the Gabor and DoG wavelets in Figure 1.

2.3. FRAME model

The original FRAME model [99, 86, 97] for texture patterns is a stationary or spatially homogeneous Markov random field model [2, 28] defined by the following probability distribution:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} \lambda_k \left([F_k * \mathbf{I}](x)\right)\right], \tag{6}$$

where each $\lambda_k()$ is a one-dimensional non-linear function to be estimated from the training data, $\lambda = (\lambda_k(), k = 1, ..., K)$, and $Z(\lambda)$ is the normalizing constant that makes $p(\mathbf{I}; \lambda)$ integrate to 1. Model (6) is stationary because the function $\lambda_k()$ does not depend on position x.

$\lambda_k()$ can be further parametrized, e.g., $\lambda_k(r) = w_k h(r)$ for some given $h()$, to make (6) an exponential family model.

As a Markov random field model or a Gibbs distribution, the FRAME model represents the potential functions in the form of $\lambda_k\left([F_k * \mathbf{I}](x)\right)$, i.e., one-dimensional non-linear transformations of filter responses. The model achieves the maximum entropy among all the distributions with fixed marginal distributions of $[F_k * \mathbf{I}](x)$ for $k = 1, ..., K$.

The filters $\{F_k\}$ can be designed, such as the Gabor filters, or be learned from the data together with λ_k.

3. Sparse FRAME

This section presents the sparse FRAME model. We start from a dense version of the model. We then present the maximum likelihood learning algorithm. After that, we describe the generative boosting algorithm for learning the sparse version of the model.

3.1. Dense model

We start from the non-stationary or spatially inhomogeneous FRAME model [87, 89, 84] based on a dictionary of basis functions or wavelets $\{B_{k,x}, \forall k, x\}$ (we assume that the dictionary of wavelets, such as the Gabor and DoG wavelets in Figure 1, has been given or has been learned by sparse component analysis [66, 4, 19]). The model is a random field of the following form:

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} w_{k,x} h(\langle \mathbf{I}, B_{k,x} \rangle)\right] q(\mathbf{I}). \tag{7}$$

The above model is a simple generalization of the FRAME model (6), where $\langle \mathbf{I}, B_{k,x} \rangle$ is the filter response, which can also be written as $[F_k * \mathbf{I}](x)$. The parameter $w_{k,x}$ depends on position x, so the model is non-stationary. $w = (w_{k,x}, \forall k, x)$. Again $Z(w)$ is the normalizing constant. $h()$ is a pre-specified rectification function. In [87], $h(r) = |r|$, i.e., the model is insensitive to the signs of filter responses. $q(\mathbf{I})$ is a reference distribution, such as the uniform distribution or the Gaussian white noise model

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left[-\frac{1}{2\sigma^2} ||\mathbf{I}||^2\right], \tag{8}$$

where D counts the number of pixels in the image domain $\mathcal{D}$.

The reference distribution $q(\mathbf{I})$ can be considered an initial null model without any features. $p(\mathbf{I}; w)$ is an exponential tilting of $q(\mathbf{I})$ to modify $q(\mathbf{I})$ to be close to the data distribution. According to the maximum entropy principle, among all the distributions p with the same expectations $\mathrm{E}_p[h(\langle \mathbf{I}, B_{k,x} \rangle)]$ for all (k, x), $p(\mathbf{I}; w)$ achieves the minimal Kullback-Leibler divergence $\mathrm{KL}(p|q)$. That is, $p(\mathbf{I}; w)$ is the minimal modification of the null model q among all the distributions that reproduce the observed feature statistics.

In the original FRAME model (6), $q(\mathbf{I})$ is the uniform distribution and is made implicit. For Gaussian white noise $q(\mathbf{I})$, the parameter σ^2 can be

either estimated from the data together with other parameters or be fixed at a certain value. For the sparse FRAME model to be reviewed later, σ^2 can be interpreted as the variance of the residual image of the sparse coding representation, and σ^2 is often set at a small value. For the dense FRAME model or the deep FRAME model to be reviewed later, the choice of σ^2 is not very critical, because the exponential tilting can be very flexible.

3.2. Maximum likelihood learning

The basic learning algorithm estimates the parameters $w = (w_{k,x}, \forall k, x)$ from a set of aligned training images $\{\mathbf{I}_i, i = 1, ..., n\}$ that come from the same category, where n is the total number of training images. The algorithm can be extended to learn from non-aligned images from mixed categories. The basic learning algorithm seeks to maximize the log-likelihood

$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \log p(\mathbf{I}_i; w), \tag{9}$$

which is a concave function, whose partial derivatives are

$$\frac{\partial L(w)}{\partial w_{k,x}} = \frac{1}{n} \sum_{i=1}^{n} h(\langle \mathbf{I}_i, B_{k,x} \rangle) - \mathrm{E}_w \left[h(\langle \mathbf{I}, B_{k,x} \rangle) \right], \tag{10}$$

where E_w denotes expectation with respect to $p(\mathbf{I}; w)$ in (7). The key to the above identify is that $\partial \log Z(w)/\partial w_{k,x} = \mathrm{E}_w \left[h(\langle \mathbf{I}, B_{k,x} \rangle) \right]$. This expectation can be approximated by Monte Carlo integration. Thus, w can be computed by the stochastic gradient ascent algorithm [72, 92]

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma_t \left[\frac{1}{n} \sum_{i=1}^{n} h(\langle \mathbf{I}_i, B_{k,x} \rangle) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} h(\langle \tilde{\mathbf{I}}_i, B_{k,x} \rangle) \right], \tag{11}$$

where γ_t is the step size or the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images sampled from $p(\mathbf{I}; w^{(t)})$ using MCMC, such as Hamiltonian Monte Carlo [63] or the Gibbs sampler [27]. $\tilde{n}$ is the total number of independent parallel Markov chains that sample from $p(\mathbf{I}; w^{(t)})$. We initialize the learning from $w^{(0)} = 0$, and the initial synthetic images are sampled from $q(\mathbf{I})$, i.e., the Gaussian white noise images. By gradually updating the parameters, the distribution of the synthetic images becomes closer to the distribution of the observed images.

3.3. Generative boosting

Model (7) is a dense model in that all the wavelets (or filters) in the dictionary are included in the model. We can sparsify the model by forcing most of the $w_{k,x}$ to be zero, so that only a small number of wavelets are included in the model. This can be achieved by a generative version [89] of the epsilon-boosting algorithm [25, 23] (see also [22, 10, 83, 85]). The algorithm starts from $w = 0$, the zero vector. At the t-th iteration, let

$$\Delta_{k,x} = \frac{1}{n}\sum_{i=1}^{n} h(\langle \mathbf{I}_i, B_{k,x}\rangle) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} h(\langle \tilde{\mathbf{I}}_i, B_{k,x}\rangle) \tag{12}$$

be the Monte Carlo estimate of $\partial L(w)/\partial w_{k,x}$, where again $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images sampled from the current model. We select

$$(\hat{k}, \hat{x}) = \arg\max_{k,x} \Delta_{k,x}, \tag{13}$$

and update $w_{\hat{k},\hat{x}}$ by

$$w_{\hat{k},\hat{x}}^{(t+1)} = w_{\hat{k},\hat{x}}^{(t)} + \gamma_t \Delta_{\hat{k},\hat{x}}, \tag{14}$$

where γ_t is the step size at the t-th step, which is assumed to be sufficiently small (thus the term "epsilon" in the epsilon-boosting algorithm). We call this algorithm generative epsilon boosting because the derivatives are estimated by images generated from the current model. See Figure 2 for an illustration. The training images are of the size 100×100, whose intensities are within $[0, 255]$. We fix $\sigma^2 = 1$ in the reference distribution q.

The selected wavelet $B_{\hat{k},\hat{x}}$ reveals the dimension along which the current model is most conspicuously lacking in reproducing the statistical properties of the training images. By including $B_{\hat{k},\hat{x}}$ into the model and updating the corresponding parameter $w_{\hat{k},\hat{x}}$, the model receives the most needed boost. The process is like an artist making a painting, where $B_{\hat{k},\hat{x}}$ is the stroke that is most needed to make the painting look more similar to the observed objects.

The epsilon boosting algorithm [25, 35] has an interesting relationship with the ℓ_1 regularization in the Lasso [78] and basis pursuit [5]. As pointed out by [73], under a monotonicity condition (e.g., the components of w keep increasing), such an algorithm approximately traces the solution path of the ℓ_1 regularized minimization of

$$-L(w) + \rho\|w\|_{\ell_1}, \tag{15}$$

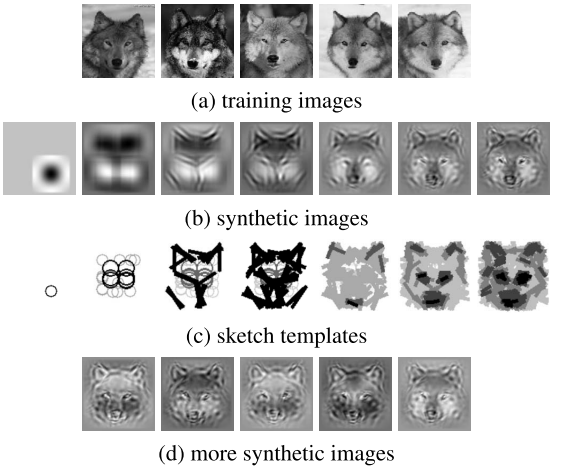


Figure 2: Learning process of the generative boosting. (a) 5 observed training images (100×100 pixels) from which the random field model is learned. (b) a sequence of synthetic images generated by the learned model as more and more wavelets are induced into the model. The numbers of the selected wavelets are $1, 20, 65, 100, 200, 500$, and 800 respectively. (c) a sequence of sketch templates that illustrate the wavelets selected from the given 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 synthetic images independently generated from the final learned model.

where the regularization parameter ρ starts from a big value so that all the components of w are zero, and gradually lowers itself to allow more components to be non-zero so that more wavelets are induced into the model.

3.4. Sparse model

After selecting m wavelets, we have the following sparse FRAME model:

$$p(\mathbf{I}; \mathbf{B}, w) = \frac{1}{Z(w)} \exp\left[\sum_{j=1}^{m} w_j h(\langle \mathbf{I}, B_{k_j,x_j} \rangle)\right] q(\mathbf{I}), \tag{16}$$

where $\mathbf{B} = (B_j = B_{k_j,x_j}, j = 1, ..., m)$ is the set of wavelets selected from the dictionary, and $w_j = w_{k_j,x_j}$.

In model (16), m is much smaller than D, the number of pixels. Thus, we can represent $\mathbf{I}$ by

$$\mathbf{I} = \sum_{j=1}^{m} c_j B_{k_j,x_j} + \epsilon, \tag{17}$$

where $C = (c_j, j = 1, ..., m)^\top$ are the least square regression coefficients of $\mathbf{I}$ on $\mathbf{B} = (B_j, j = 1, ..., m)$, i.e., $C = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{I}$, and ϵ is the residual image. The distribution of C under $p(\mathbf{I}; \mathbf{B}, w)$ is

$$p_C(C; w) = \frac{1}{Z(w)} \exp\left[\langle w, h(\mathbf{B}^\top\mathbf{B}C)\rangle\right] q_C(C), \tag{18}$$

where $q_C(C)$ is the distribution of C under $q(\mathbf{I})$, and the transformation $h()$ is applied element-wise. Thus, $p(\mathbf{I}; B, w)$ in (16) can be written as a wavelet sparse coding model (17) and (18). The forms of (16) and (17) show that the selected wavelets $\{B_j\}$ serve as both filters and basis functions. The sparse coding form of the model (17) and (18) is used for sampling $\{\tilde{\mathbf{I}}_i\}$ from $p(\mathbf{I}; \mathbf{B}, w)$ by first sampling $C \sim p_C(C; w)$ using the Gibbs sampler [27], and then generating $\tilde{\mathbf{I}}_i$ according to (17).

Model (17) suggests that we can also select the wavelets by minimizing

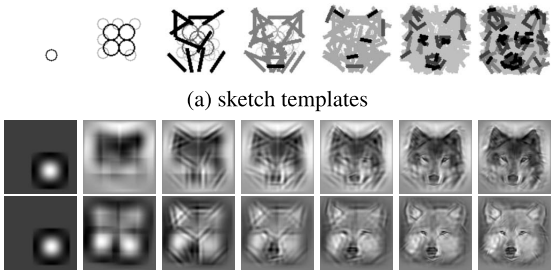
$$\sum_{i=1}^{n} \|\mathbf{I}_i - \sum_{j=1}^{m} c_{i,j} B_{k_j,x_j}\|^2, \tag{19}$$

using a shared matching pursuit method [87]. See Figure 3 for an illustration. We can also allow the selected wavelets to perturb their locations and orientations to account for shape deformations [84].

The sparse FRAME model can be used for unsupervised learning tasks such as model-based clustering [21]. Extending the learning algorithm, one can learn a codebook of multiple sparse FRAME models from non-aligned images. The learned models can be used for tasks such as transfer learning [87, 41].

The sparse FRAME model merges two important research themes in image representation and modeling, namely, Markov random fields [2, 28] and wavelet sparse coding [66, 19].

The wavelets can be mapped to the first layer filters of a ConvNet [52] to be described below. The sparse FRAME models can be mapped to the second layer nodes of a ConvNet, except that the sparse FRAME versions of the second layer nodes are selectively and sparsely connected to the first layer nodes.



(a) sketch templates

(b) reconstructed images

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

4. Deep FRAME

In the deep FRAME model, the filters are non-linear filters in a pre-trained ConvNet. We shall first review the ConvNet and then present the deep FRAME model.

4.1. ConvNet

The convolutional neural network (CNN or ConvNet) [52] is a specialized neural network devised for analyzing signals such as images, where the linear transformations take place around each pixel, i.e., they are filters or convolutions. See Figure 4 for an illustration.

A ConvNet consists of multiple layers of linear filtering and point-wise non-linear transformation, as expressed by the following recursive formula:

$$[F_j^{(l)} * \mathbf{I}](y) = h\left(\sum_{k=1}^{N_{l-1}} \sum_{x \in \mathcal{S}_l} w_{k,x}^{(l,j)} [F_k^{(l-1)} * \mathbf{I}](y+x) + b_{l,j}\right), \tag{20}$$

or

$$\mathbf{I}_j^{(l)}(y) = h\left(\sum_{k=1}^{N_{l-1}} \sum_{x \in \mathcal{S}_l} w_{k,x}^{(l,j)} \mathbf{I}_k^{(l-1)}(y+x) + b_{l,j}\right), \tag{21}$$

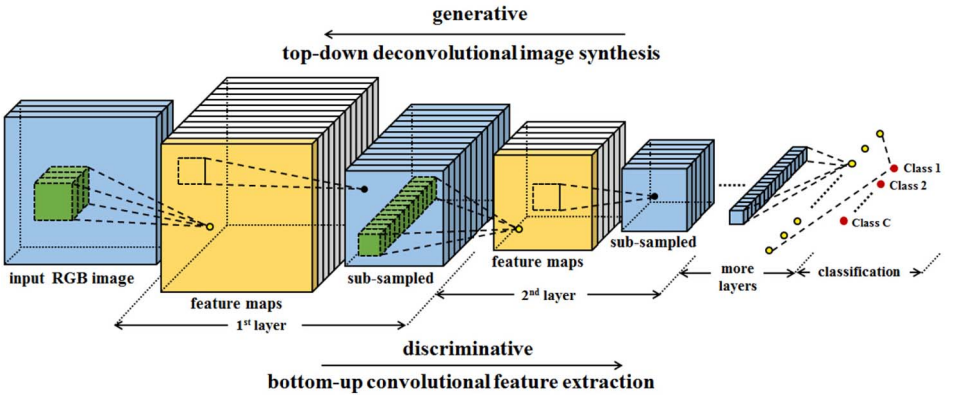


Figure 4: Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image.

where $l = 1, ..., L$ indexes the layer, and $\mathbf{I}_j^{(l)} = F_j^{(l)} * \mathbf{I}$ are filtered images or feature maps at layer l. In Figure 4, the feature maps are illustrated by the square shapes. Each $[F_j^{(l)} * \mathbf{I}](x)$ is called a filter response or a feature extracted by a node or a unit at layer l.

$\{F_j^{(l)}, j = 1, ..., N_l\}$ are the filters at layer l, and $\{F_k^{(l-1)}, k = 1, ..., N_{l-1}\}$ are the filters at layer $l-1$. j and k are used to index the filters at layers l and $l-1$ respectively, and N_l and N_{l-1} are the numbers of filters at layers l and $l-1$ respectively. The filters are locally supported, so the range of x in $\sum_x$ is within a local support $\mathcal{S}_l$ (such as a 7×7 image patch). We let $\mathbf{I}^{(0)} = \mathbf{I}$. The filter responses at layer l are computed from the filter responses at layer $l-1$, by linear filtering defined by the weights $w_{k,x}^{(l,j)}$ as well as the bias term $b_{l,j}$, followed by the non-linear transformation $h()$. The most commonly used non-linear transformation in the modern ConvNets is the rectified linear unit (ReLU) [50],

$$h(r) = \max(0, r). \tag{22}$$

$\{F_j^{(l)}\}$ are non-linear filters because we incorporate $h()$ in the computation of the filter responses. We call $\mathbf{I}_j^{(l)} = F_j^{(l)} * \mathbf{I}$ the filtered image or the feature map of filter j at layer l. We denote $\mathbf{I}^{(l)} = (\mathbf{I}_j^{(l)}, j = 1, ..., N_l)$, which consists

of a total of N_l feature maps at layer l, and $j = 1, ..., N_l$. Sometimes, people call $\mathbf{I}^{(l)}$ as a whole feature map or filter image with N_l channels, where each $\mathbf{I}_j^{(l)}$ corresponds to one channel. For a colored image, $\mathbf{I}^{(0)} = \mathbf{I}$ has 3 channels for RGB.

The filtering operations are often followed by sub-sampling and local-max pooling (e.g., $\mathbf{I}(x_1, x_2) \leftarrow \max_{(s_1,s_2)\in\{0,1\}^2} \mathbf{I}(2x_1 + s_1, 2x_2 + s_2)$). See Figure 4 for an illustration of sub-sampling. After a number of layers with sub-sampling, the filtered images or feature maps are reduced to 1×1 at the top layer. These features are then used for classification (e.g., does the image contain a hummingbird or a seagull or a dog) via multinomial logistic regression.

4.2. FRAME with ConvNet filters

Instead of using linear filters as in the original FRAME model, we can use the filters at a certain convolutional layer of a pre-learned ConvNet. We call such a model the deep FRAME model.

Suppose there exists a bank of filters $\{F_k^{(l)}, k = 1, ..., K\}$ at a certain convolutional layer l of a pre-learned ConvNet, as recursively defined by (20). For an image $\mathbf{I}$ defined on the image domain $\mathcal{D}$, let $F_k^{(l)} * \mathbf{I}$ be the feature map of filter $F_k^{(l)}$, and let $[F_k^{(l)} * \mathbf{I}](x)$ be the filter response of $\mathbf{I}$ to $F_k^{(l)}$ at position x (again x is a two-dimensional coordinate). We assume that $[F_k^{(l)} * \mathbf{I}](x)$ is the response obtained after applying the non-linear transformation or rectification function $h()$ in (22). Then the non-stationary deep FRAME model becomes

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K} \sum_{x\in\mathcal{D}} w_{k,x}[F_k^{(l)} * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{23}$$

where $q(\mathbf{I})$ is again the Gaussian white noise model (8), and $w = (w_{k,x}, \forall k, x)$ are the unknown parameters to be learned from the training data. Model (23) shares the same form as model (7) with linear filters, except that the rectification function $h(r) = \max(0, r)$ in model (7) is already absorbed in the ConvNet filters $\{F_k^{(l)}\}$ in model (23). We can also make model (23) stationary by letting $w_{k,x} = w_k$ for all x.

4.3. Learning and sampling

The basic learning algorithm estimates the unknown parameters w from a set of aligned training images $\{\mathbf{I}_i, i = 1, ..., n\}$ that come from the same

object category. Again the weight parameters w can be estimated by maximizing the log-likelihood function, which is a concave function, and w can be computed by the stochastic gradient ascent algorithm [92]:

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma_t \left[\frac{1}{n} \sum_{i=1}^{n} [F_k^{(l)} * \mathbf{I}_i](x) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} [F_k^{(l)} * \tilde{\mathbf{I}}_i](x) \right] \tag{24}$$

for every $k \in \{1, ..., K\}$ and $x \in \mathcal{D}$, where γ_t is the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images sampled from $p(\mathbf{I}; w^{(t)})$ using MCMC. This is an analysis by synthesis scheme that seeks to match the average filter responses of the synthetic images to those of the observed images.

In order to sample from $p(\mathbf{I}; w)$, we adopt the Langevin dynamics [55, 30]. Writing the energy function

$$U(\mathbf{I}, w) = -\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} w_{k,x} [F_k^{(l)} * \mathbf{I}](x) + \frac{1}{2\sigma^2} ||\mathbf{I}||^2, \tag{25}$$

the Langevin dynamics iterates

$$\mathbf{I}_{\tau+1} = \mathbf{I}_\tau - \delta U'(\mathbf{I}_\tau, w) + \sqrt{2\delta} \epsilon_\tau, \tag{26}$$

where $U'(\mathbf{I}, w) = \partial U(\mathbf{I}, w)/\partial \mathbf{I}$. This gradient can be computed by back-propagation. In (26), δ is a small step-size, and $\epsilon_\tau \sim \mathrm{N}(0, I_D)$, independently across τ, where I_D is the identity matrix of dimension $D = |\mathcal{D}|$, i.e., the dimensionality of $\mathbf{I}$. ϵ_τ is a Gaussian white noise image whose pixel values follow $\mathrm{N}(0, 1)$ independently. Here we use τ to denote the time steps of the Langevin sampling process, because t is used for the time steps of the learning process. The Langevin sampling process (26) is an inner loop within the learning process (24). Between every two consecutive updates of w in the learning process, we run a finite number of steps of the Langevin dynamics starting from the images generated by the previous iteration of the learning algorithm.

The Langevin dynamics was first applied to the FRAME model by [97], where the gradient descent component is interpreted as the Gibbs Reaction And Diffusion Equation (GRADE), and the patterns are formed via the reactions and diffusions controlled by different types of filters.

Again we initialize the learning algorithm from $w^{(0)} = 0$, and the initial synthesized images are sampled from $q(\mathbf{I})$, i.e., the white noise images.

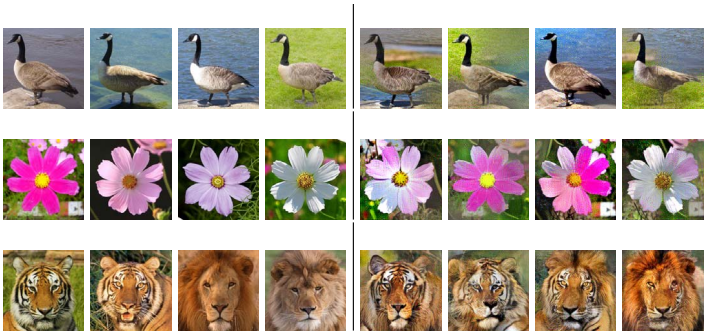


Figure 5: Generating object patterns. In each row, the left half displays 4 of the training images (224×224), and the right half displays 4 of the synthetic images. In the last row, the learned model generates hybrid patterns of lion and tiger.

We first learn a non-stationary FRAME model (23) from images of aligned objects of the same pose. The images were collected from the internet. For each category, the number of training images was around 10. We used $\tilde{n} = 16$ parallel chains for Langevin sampling with 100 Langevin steps between every two consecutive updates of the parameters. Figure 5 shows some experiments using filters from the 3rd convolutional layer of the VGG ConvNet [76], a commonly used pre-learned ConvNet trained on Imagenet ILSVRC2012 dataset [12]. For each experiment on each row, the left half displays 4 of the training images, and the right half displays 4 of the synthetic images generated by the Langevin dynamics. The last experiment is about learning the hybrid pattern of lion and tiger. The model re-mixes local image patterns seamlessly.

Figure 6 shows results from experiments on the stationary model for texture images. The model does not require image alignment. It re-shuffles the local patterns seamlessly. Each experiment is illustrated by 3 images, where the first image is the training image, and the other 2 images are generated by the learning algorithm. In the last 3 images, the first 2 images are training images, and the last image is generated by the learned model that mixes the patterns of brick wall and ivy.

4.4. Learning a new layer of filters

On top of the existing pre-learned convolutional layer of filters $\{F_k^{(l)}, k = 1, ..., K\}$, we can build another layer of filters $\{F_j^{(l+1)}, j = 1, ..., J\}$, according

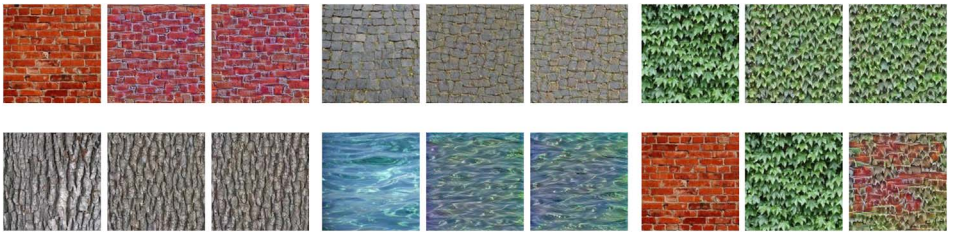


Figure 6: Generating texture patterns. For each category, the first image (224 $\times$ 224) is the training image, and the next 2 images are generated images, except for the last 3 images, where the first 2 are the training images, and the last one is the generated image that mixes brick wall and ivy.

to the recursive formula (20), so that

$$[F_j^{(l+1)} * \mathbf{I}](y) = h\left(\sum_{k,x} w_{k,x}^{(j)} [F_k^{(l)} * \mathbf{I}](y+x) + b_j\right), \tag{27}$$

where $h(r) = \max(0, r)$. The set $\{F_j^{(l+1)}\}$ is like a dictionary of "words" to describe different types of objects or patterns in the training images.

Due to the recursive nature of ConvNet, the deep FRAME model (23) based on filters $\{F_k^{(l)}\}$ corresponds to a single filter in $\{F_j^{(l+1)}\}$ at a particular position y (e.g., the origin $y = 0$) where we assume that the object appears. In [8], we show that the rectification function $h(r) = \max(0, r)$ can be justified by a mixture model where the object can either appear at a position or not. The bias term is related to $-\log Z(w)$.

Model (23) is used to model images where the objects are aligned and are of the same category. For images of non-aligned objects from multiple categories, we can extend model (23) to a convolutional version with a whole new layer of multiple filters

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{j=1}^{J} \sum_{x \in \mathcal{D}} [F_j^{(l+1)} * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{28}$$

where $\{F_j^{(l+1)}\}$ are defined by (27). This model is a product of experts model [38, 74], where each $[F_j^{(l+1)} * \mathbf{I}](x)$ is an expert about a mixture of an activation or inactivation of an object of type j at position x. The stationary model for textures (in Figure 6) is a special case of this model.

Suppose we observe images of non-aligned patterns $\{\mathbf{I}_i, i = 1, ..., n\}$, and we want to learn a new layer of filters $\{F_j^{(l+1)}, j = 1, ..., J\}$ by fitting the model (28) with (27) to the observed images, where $\{F_j^{(l+1)}\}$ model different types of patterns in these images. This is an unsupervised learning problem because we do not know where the patterns are. The model can still be learned by the analysis by synthesis scheme as before.

Let $L(w) = \frac{1}{n}\sum_{i=1}^{n} \log p(\mathbf{I}_i; w)$ be the log-likelihood where $p(\mathbf{I}; w)$ is defined by (28) and (27). Then the gradient ascent learning algorithm is based on

$$\begin{aligned} \frac{\partial L(w)}{\partial w_{k,x}^{(j)}} &= \frac{1}{n}\sum_{i=1}^{n}\sum_{y\in\mathcal{D}} s_{j,y}(\mathbf{I}_i)[F_k^{(l)} * \mathbf{I}_i](y+x) \\ &\quad - \mathrm{E}_w\left[\sum_{y\in\mathcal{D}} s_{j,y}(\mathbf{I})[F_k^{(l)} * \mathbf{I}](y+x)\right], \end{aligned} \tag{29}$$

where

$$s_{j,y}(\mathbf{I}) = h'\left(\sum_{k,x} w_{k,x}^{(j)}[F_k^{(l)} * \mathbf{I}](y+x) + b_j\right) \tag{30}$$

is a binary on/off detector of object j at position y on image $\mathbf{I}$, because for $h(r) = \max(0, r)$, $h'(r) = 0$ if $r \le 0$, and $h'(r) = 1$ if $r > 0$. The gradient (29) admits an EM [11] interpretation which is typical in unsupervised learning algorithms that involve hidden variables. Specifically, $s_{j,y}()$ detects the pattern of type j that is modeled by $F_j^{(l+1)}$ at location y. This step can be considered a hard-decision E-step. With the patterns detected, the parameters of $F_j^{(l+1)}$ are then refined in a similar way as in (24), which can be considered the M-step. That is, we learn $F_j^{(l+1)}$ only from image patches where patterns of type j are detected.

For this model as well as the models in the subsequent sections, the log-likelihood is not concave anymore, thus the maximum likelihood learning algorithm will converge to a local maximum. Adopting the common practice of training neural networks, we initialize the learning algorithm from small parameter values sampled from a Gaussian white noise distribution with small variance, and update the parameters by stochastic gradient ascent. The synthesized images are again initialized from the Gaussian white noise distribution $q(\mathbf{I})$.

Figure 7: Learning a new layer of filters without requiring object bounding boxes or image alignment. For each experiment, the first image (224×224) is the training image, and the next 2 images are generated by the learned model.

Figure 7 displays two experiments. In each experiment, the first image (224×224) is the training image, and the rest 2 images are generated by the learned model. In the first scenery experiment, we learn 10 filters at the 4th convolutional layer, based on the pre-trained VGG filters at the 3rd layer. The size of each Conv4 filter to be learned is $11 \times 11 \times 256$. In the second sunflower experiment, we learn 20 filters of size $7 \times 7 \times 256$. Clearly these learned filters capture the local patterns and re-shuffle them seamlessly.

4.5. Deep convolutional energy-based model

Instead of relying on the pre-trained filters from an existing ConvNet, we can also learn the filters $\{F_k^{(l)}, k = 1, ..., K\}$ themselves. The resulting model is a deep convolutional energy-based model [65, 8, 90],

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp[f(\mathbf{I}; w)] q(\mathbf{I}), \tag{31}$$

where $f(\mathbf{I}; w)$ is defined by a ConvNet. In model (28) with (27), we have

$$f(\mathbf{I}; w) = \sum_{j=1}^{J} \sum_{x \in \mathcal{D}} [F_j^{(l+1)} * \mathbf{I}](x). \tag{32}$$

Using more compact notation, we can define $f(\mathbf{I}; w)$ recursively by

$$\mathbf{I}^{(l)} = h(w_l \mathbf{I}^{(l-1)} + b_l), \tag{33}$$

for $l = 1, ..., L$, where $h()$ is applied element-wise. $\mathbf{I}^{(0)} = \mathbf{I}$, and $f(\mathbf{I}; w) = \mathbf{I}^{(L)}$. $\mathbf{I}^{(l)}$ consists of all the filtered images or feature maps at layer l, and the rows of w_l consist of all the filters as well as all the locations where the

filters operate on $\mathbf{I}^{(l-1)}$ to extract the features in $\mathbf{I}^{(l)}$. We assume that at the final layer L, $\mathbf{I}^{(L)}$ is reduced to a number (i.e., a 1×1 feature map). $w = (w_l, b_l, l = 1, ..., L)$. We can compare the compact equation (33) with the more detailed equation (21).

For piecewise linear $h()$, such as $h(r) = \max(0, r)$, the function $f(\mathbf{I}; w)$ is piecewise linear [68, 61]. Specifically, $h(r) = \max(0, r) = 1(r > 0)r$, where $1(r > 0)$ is the indicator function that returns 1 if $r > 0$ and 0 otherwise. Then

$$\mathbf{I}^{(l)} = s_l(\mathbf{I}; w)(w_l \mathbf{I}^{(l-1)} + b_l), \tag{34}$$

where

$$s_l(\mathbf{I}; w) = \text{diag}(1(w_l \mathbf{I}^{(l-1)} + b_l > 0)), \tag{35}$$

i.e., a diagonal matrix of binary indicators (the indicator function is applied element-wise) [68]. Let $s = (s_l, l = 1, ..., L)$ consists of indicators at all the layers, then

$$f(\mathbf{I}; w) = \mathbf{B}_{s(\mathbf{I};w)} \mathbf{I} + a_{s(\mathbf{I};w)} \tag{36}$$

is piecewise linear, where

$$\mathbf{B}_s = \prod_{l=L}^{1} s_l w_l, \tag{37}$$

and a_s can be similarly calculated. $s(\mathbf{I}; w)$ partitions the image space of $\mathbf{I}$ into exponentially many pieces [68] according to the value of $s(\mathbf{I}; w)$. The partition is recursive because $s_l(\mathbf{I}; w)$ depends on $s_{l-1}(\mathbf{I}; w)$. The boundaries between the pieces are all linear. On each piece with $s(\mathbf{I}; w) = s$, where s on the right-hand side denotes a particular instantiation of $s(\mathbf{I}; w)$, $f(\mathbf{I}; w)$ is a linear function $f(\mathbf{I}; w) = \mathbf{B}_s \mathbf{I} + a_s$. The binary switches in $s(\mathbf{I}; w)$ reconfigure the linear transformation according to (37).

$f(\mathbf{I}; w)$ generalizes three familiar structures in statistics:

(1) Generalized linear model (GLM). A GLM structure is a composition of a linear combination of the input variables and a non-linear link function. A ConvNet can be viewed as a recursion of this structure, where each component of $\mathbf{I}^{(l)}$ is a GLM transformation of $\mathbf{I}^{(l-1)}$, with h being the link function.

(2) Linear spline. A one-dimensional linear spline is of the form $y = \beta_0 + \sum_{k=1}^{d} \beta_k \max(0, x - a_k)$, where a_k are the knots. The ConvNet $f(\mathbf{I}; w)$

can be viewed as a multi-dimensional linear spline. The number of linear pieces is exponential in the number of layers [68]. Such a structure can approximate any continuous non-linear function by a large number of linear pieces.

(3) CART [3] and MARS [24]. In the classification and regression tree (CART) and the multivariate adaptive regression splines (MARS), the input domain is recursively partitioned. The linear pieces mentioned above are also recursively partitioned according to the values of $s_l(\mathbf{I}; w)$ for $l = 1, ..., L$. Moreover, MARS also makes use of the hinge function $\max(0, r)$.

For Gaussian reference $q(\mathbf{I})$, the energy function is

$$U(\mathbf{I}; w) = -f(\mathbf{I}; w) + \frac{1}{2\sigma^2}||\mathbf{I}||^2. \tag{38}$$

We can continue to use Langevin dynamics (26) to sample from $p(\mathbf{I}; w)$.

The parameter w can be learned by the stochastic gradient ascent algorithm [92]

$$w^{(t+1)} = w^{(t)} + \gamma_t \left[\frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} f(\mathbf{I}_i; w) - \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial w} f(\tilde{\mathbf{I}}_i; w) \right], \tag{39}$$

where again γ_t is the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images sampled from $p(\mathbf{I}; w^{(t)})$. This is again an analysis by synthesis scheme. This step shifts the probability density function $p(\mathbf{I}; w)$, or more specifically, the high probability regions or the low energy regions, from the synthetic images $\{\tilde{\mathbf{I}}_i\}$ to the observed images $\{\mathbf{I}_i\}$.

In the sampling step, we need to compute $\partial f(\mathbf{I}; w)/\partial \mathbf{I}$. In the learning step, we need to compute $\partial f(\mathbf{I}; w)/\partial w$. Both derivatives can be calculated by the chain rule back-propagation, and they share the computations of $\partial \mathbf{I}^{(l)}/\partial \mathbf{I}^{(l-1)}$.

Our experiments show that the model is quite expressive. For example, we learn a 3-layer model. The first layer has 100 15×15 filters with sub-sampling size of 3 pixels. The second layer has 64 5×5 filters with sub-sampling size of 1. The third layer has 30 3×3 filters with sub-sampling size of 1. We learn a model (31) for each texture category from a single training image. Figure 8 displays some results. For each category, the first image is the training image, and the rest are 2 of the images generated by the learning algorithm. We use $\tilde{n} = 16$ parallel chains for Langevin sampling. The number of Langevin iterations between every two consecutive updates of parameters is 10. The training images are of the size 224×224, whose intensities are within $[0, 255]$. We fix $\sigma^2 = 1$ in the reference distribution q.

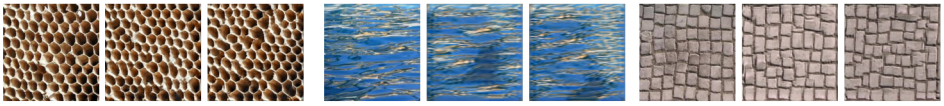


Figure 8: Generating texture patterns. For each category, the first image (224 $\times$ 224) is the training image, and the rest are 2 of the images generated by the learning algorithm.

Figure 9: Generating object patterns. For each category, the left panel displays 100 randomly sampled training image, and the right panel displays 100 randomly sampled synthesized images.

In our recent work [26], we develop a multi-grid modeling and sampling method for learning the deep convolutional energy-based model. Figure 9 displays the results of two experiments, where in each row, the left panel consists of randomly sampled training images, and the right panel consists of

randomly sampled synthetic images generated by the learned model. In the first experiment, we learn the model from 10,000 images randomly sampled from the CelebA [56] dataset of face images, and the image size is 64×64. In the second experiment, we learn the model from 73,257 training images from the SVHN dataset [64] of house numbers collected by Google Street View. The learned models can be used for classification and pattern completion. See [26] for details.

5. Hopfield auto-encoder

Consider the sparse FRAME model (16). Let us assume that the reference distribution $q(\mathbf{I})$ is white noise with mean 0 and variance $\sigma^2 = 1$. The energy function is

$$U(\mathbf{I}) = \frac{1}{2}\|\mathbf{I}\|^2 - \sum_{j=1}^{m} w_j h(\langle \mathbf{I}, B_{k_j,x_j} \rangle). \tag{40}$$

This energy function can be multi-modal, and each local minimum $\hat{\mathbf{I}}$ satisfies $U'(\hat{\mathbf{I}}) = 0$, which implies

$$\hat{\mathbf{I}} = \sum_{j=1}^{m} w_j h'(\langle \hat{\mathbf{I}}, B_{k_j,x_j} \rangle) B_{k_j,x_j}. \tag{41}$$

This reveals an auto-encoder [81, 31] hidden in the local modes:

$$\text{Encoding} : c_j = w_j h'(\langle \hat{\mathbf{I}}, B_{k_j,x_j} \rangle), \tag{42}$$

$$\text{Decoding} : \hat{\mathbf{I}} = \sum_{j=1}^{m} c_j B_{k_j,x_j}, \tag{43}$$

where (42) encodes $\hat{\mathbf{I}}$ by (c_j), and (43) reconstructs $\hat{\mathbf{I}}$ from (c_j). B_{k_j,x_j} serves as both bottom-up filter in (42) and top-down basis function in (43). We call this auto-encoder the Hopfield auto-encoder because $\hat{\mathbf{I}}$ is a local minimum of the energy function (40). Hopfield [42] proposes that the local energy minima may be used for content-addressable memory.

The Hopfield auto-encoder also presents itself in the deep convolutional energy-based model (31) [90]. The energy function of the model is $\|\mathbf{I}\|^2/2 - f(\mathbf{I}; w)$. The local minima satisfies the Hopfield auto-encoder $\hat{\mathbf{I}} = f'(\hat{\mathbf{I}}; w)$, or more specifically,

$$\text{Encoding} : s = s(\hat{\mathbf{I}}; w), \tag{44}$$

$$\text{Decoding} : \hat{\mathbf{I}} = \mathbf{B}_s, \tag{45}$$

where $s(\hat{\mathbf{I}}; w)$ and $\mathbf{B}_s$ are defined by (35) and (37) respectively. The encoding process is a bottom-up computation of the indicators at different layers $s_l = s_l(\mathbf{I}; w)$, for $l = 1, ..., L$, where w_l plays the role of filters, see equation (35). The decoding process is a top-down computation for reconstruction, where s_l plays the role of coefficients, and w_l plays the role of basis functions. See equation (37). The encoding process detects the patterns corresponding to the filters, and the decoding process reconstructs the image using the detected filters as the basis functions.

The relationship between auto-encoders and energy-based models [53] has been investigated by [80] and [77] for the restricted Boltzmann machine and its extensions [37]. A regularized auto-encoder is a special form of score matching estimator [44]. The Hopfield auto-encoder was first elucidated by [90].

In order to learn the parameters from training images, we may fit the Hopfield auto-encoder using the least squares reconstruction loss. After learning by auto-encoder, we may use MCMC-based learning to further refine the learning results, i.e., learn to synthesize after learning to reconstruct.

6. Generator as a sampler

In order to learn the deep FRAME model (23) or the deep convolutional energy-based model (31), we need to sample synthesized images from the current model using MCMC such as Langevin dynamics in the analysis by synthesis scheme. This is often time consuming. We can recruit a generator model [32] as a much more efficient sampler that generate synthesized images via non-iterative direct ancestral sampling.

6.1. Generator model

The generator model can be considered a non-linear multi-layer generalization of the factor analysis model. It has the following form

$$\begin{aligned} &X \sim \mathrm{N}(0, I_d); \\ &\tilde{\mathbf{I}} = g(X; \tilde{w}) + \epsilon; \epsilon \sim \mathrm{N}(0, \sigma^2 I_D). \end{aligned} \tag{46}$$

where X consists of d latent factors that follow $\mathrm{N}(0, 1)$ independently, and the image $\tilde{\mathbf{I}}$ is obtained by a top-down ConvNet that transforms X to $\tilde{\mathbf{I}}$. We use the notation $\tilde{\mathbf{I}}$ to emphasize the fact that the generator model is used to generate the synthetic images, and we use $\tilde{w}$ to denote the parameters of this model for synthetic images. To generate $\tilde{\mathbf{I}}$, we can simply generate X from its known prior distribution $\mathrm{N}(0, I_d)$, and then transform X to $\tilde{\mathbf{I}}$ by

$g(X; \tilde{w})$ plus the white noise ϵ. This is called ancestral sampling, which is non-iterative and does not require MCMC. The prior distribution $\mathrm{N}(0, I_d)$ is the same as the original factor analysis, which assumes that the components of X are the latent factors that generate $\tilde{\mathbf{I}}$, and these factors do not need further explanation as they are independent of each other.

$g(X; \tilde{w})$ can be considered a recursion of factor analysis, with

$$X^{(l-1)} = h(\tilde{w}_l X^{(l)} + \tilde{b}_l) \tag{47}$$

for $l = 1, ..., L$, and $\tilde{w} = (\tilde{w}_l, \tilde{b}_l, l = 1, ..., L)$. $\tilde{\mathbf{I}} = X^{(0)} = g(X; \tilde{w})$, and $X^{(L)} = X$. $X^{(l)}$ can be interpreted as factors at layer l. Again h is a non-linear rectification function such as $h(r) = \max(0, r)$ that is applied element-wise. In this case, $g(X; \tilde{w})$ is piecewise linear, and the model becomes a piecewise linear factor analysis.

$g(X; \tilde{w})$ is a top-down ConvNet [93, 18], which should be contrasted with $f(\mathbf{I}; w)$ in the deep FRAME model or deep convolutional energy-based model, which is a bottom-up ConvNet, as illustrated by the following diagram:

$$\begin{array}{cc} \text{Top-down ConvNet} & \text{Bottom-up ConvNet} \\ \text{latent factors} & \text{features} \\ \Downarrow & \Uparrow \\ \text{image} & \text{image} \\ \text{(a) } g(X; \tilde{w}) & \text{(b) } f(\mathbf{I}; w) \end{array} \tag{48}$$

In the literature, the generator model is trained by methods that involve learning extra networks [32, 69, 49, 71, 60]. [34] proposes an alternating back-propagation algorithm for learning the model from training images $\{\mathbf{I}_i, i = 1, ..., n\}$ without relying on an extra network. Specifically, let $q(X)$ be the prior distribution of X, and let $q(\mathbf{I}|X, \tilde{w})$ be the conditional distribution of $\mathbf{I}$ given X. Then the marginal distribution of $\mathbf{I}$ is $q(\mathbf{I}; \tilde{w}) = \int q(X) q(\mathbf{I}|X, \tilde{w}) dX$. The log-likelihood is $L(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} \log q(\mathbf{I}_i; \tilde{w})$, whose gradient can be computed based on the following identity that underlies the EM algorithm [11]

$$\frac{\partial}{\partial \tilde{w}} \log q(\mathbf{I}; \tilde{w}) = \mathrm{E}_{q(X|\mathbf{I}, \tilde{w})} \left[\frac{\partial}{\partial \tilde{w}} \log q(\mathbf{I}|X, \tilde{w}) \right], \tag{49}$$

where the expectation is with respect to the posterior distribution $q(X|\mathbf{I}, \tilde{w})$, and it can be approximated by Monte Carlo samples from $q(X|\mathbf{I}, \tilde{w})$. This leads to the following stochastic gradient descent algorithm [72, 92], which iterates the following two steps. (1) Inferring the latent factors X_i from $\mathbf{I}_i$ for each i, given the current $\tilde{w}$, by sampling from the posterior distribution

$q(X_i|\mathbf{I}_i, \tilde{w})$ using the Langevin dynamics. (2) Updating $\tilde{w}$ by gradient descent on $\sum_{i=1}^{n} \|\mathbf{I}_i - g(X_i; \tilde{w})\|^2$. In Step (1), we can sample multiple copies of X_i to approximate the expectation. Step (1) requires the computation of $\partial g(X; \tilde{w})/\partial X$, while step (2) requires the computation of $\partial g(X; \tilde{w})/\partial \tilde{w}$. Both computations can be carried out by back-propagation, and the whole algorithm is in the form of alternating back-propagation.

Our experiments show that the generator model (learned by alternating back-propagation) is very expressive in that it can generate realistic images, sounds and videos. We adopt the structure of the generator network of [69, 18]. The network $g(X; \tilde{w})$ has 5 layers of convolution with 5×5 kernels (i.e., linear superposition with 5×5 basis functions), with an up-sampling factor of 2 at each layer (i.e., the basis functions are 2 pixels apart). The number of channels in the first layer is 512 (i.e., 512 translation invariant basis functions), and is decreased by a factor 2 at each layer. There is a fully connected layer under the latent factors X. The images are of the size 64×64.

In the first experiment, we learn a model where X has two components, i.e., $X = (x_1, x_2)$, and $d = 2$. The training data are 11 images of 6 tigers and 5 lions. After training the model, we generate images using the learned top-down ConvNet for $(x_1, x_2) \in [-2, 2]^2$, where we discretize both x_1 and x_2 into 9 equally spaced values. The first panel of Figure 10 displays the synthetic images on the 9×9 panel.

In the second experiment, we learn a model with $d = 100$ from 1,000 face images randomly selected from the CelebA dataset [56]. The middle panel of Figure 10 displays the images generated by the learned model, where for each synthetic image, we first generate $X \sim \mathrm{N}(0, I_d)$, and then transform it to the synthetic image by the learned network $g(X; \tilde{w})$. The right panel displays the interpolation results. The images at the four corners are generated by the inferred X vectors of four images randomly selected from the training set, where for each selected image $\mathbf{I}$, we infer X by sampling from $q(X|\mathbf{I}, \tilde{w})$. The images in the middle are obtained by first interpolating the inferred X's of the four corner images using the sphere interpolation [16] and then generating the images by the learned network.

6.2. Cooperative learning

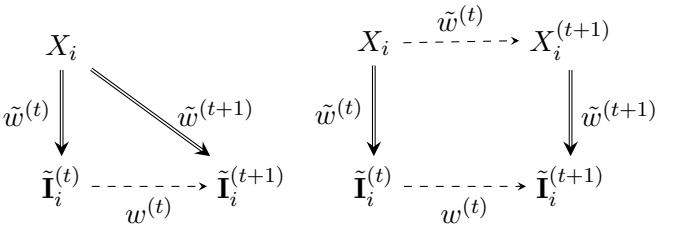
The challenge in learning the generator model from the observed images is that for each observed image $\mathbf{I}_i$, the latent factors in X_i are unknown, and must be inferred. The inference of X_i requires expensive MCMC such as Langevin dynamics. The learning is called unsupervised because X_i is not given.

Figure 10: Modeling object patterns. Left: The 64×64 synthetic images are generated by $g(X; \tilde{w})$ with the learned $\tilde{w}$, where $X = (x_1, x_2) \in [-2, 2]^2$, and X is discretized into 9×9 values. Middle: Each synthetic image is generated by first sampling $X \sim \mathrm{N}(0, I_{100})$ and then generating the image by $g(X; \tilde{w})$ with the learned $\tilde{w}$. Right: Interpolation. The images at the four corners are reconstructed from the inferred X vectors of four images randomly selected from the training set. Each image in the middle is obtained by first interpolating the X vectors of the four corner images, and then generating the image by $g(X; \tilde{w})$.

[88] proposes a cooperative learning algorithm that incorporates the generator model into the learning of the deep FRAME model or the deep convolutional energy-based model (see also [48]). The basic idea is that we still learn the deep FRAME model (23) (or the deep convolutional energy-based model (31)) via the analysis by synthesis scheme. However, we recruit the generator model to jumpstart the MCMC sampling such as the Langevin dynamics that samples from the deep FRAME model, because it is much easier to generate synthetic images from the generator model via direct ancestral sampling. Meanwhile, we let the generator model learn from the synthetic images, in particular, how the MCMC changes the synthesized images.

The following diagrams explain the basic idea:

(50)



The diagram on the left illustrates a simple learning scheme. In each iter-

ation, we generate X_i from its known prior distribution $\mathrm{N}(0, I_d)$. Then we generate $\tilde{\mathbf{I}}_i^{(t)} \sim g(X_i; \tilde{w}^{(t)}) + \epsilon_i$ according to the current generator model with parameter $\tilde{w}^{(t)}$, for $i = 1, ..., \tilde{n}$. After that, we initialize the MCMC such as the Langevin dynamics from $\tilde{\mathbf{I}}_i^{(t)}$, and run a finite number of steps of MCMC to get $\tilde{\mathbf{I}}_i^{(t+1)}$ by sampling from the current deep FRAME model with parameter $w^{(t)}$. We then update the deep FRAME model to $w^{(t+1)}$ based on $\{\tilde{\mathbf{I}}_i^{(t+1)}, i = 1, ..., \tilde{n}\}$ according to (39). Meanwhile, we update the generator model to $\tilde{w}^{(t+1)}$ by gradient descent on

$$\sum_{i=1}^{\tilde{n}} \|\tilde{\mathbf{I}}_i^{(t+1)} - g(X_i; \tilde{w})\|^2, \tag{51}$$

over $\tilde{w}$. In the above scheme, the generator model learns from the synthetic images $\{\tilde{\mathbf{I}}_i\}$, where for each $\tilde{\mathbf{I}}_i$, the latent factors X_i are known, so that there is no need to infer X_i, and the learning becomes a much simpler supervised learning problem. The diagram on the right of (50) illustrates a more rigorous scheme, where we sample $X_i^{(t+1)}$ from the posterior distribution $q(X_i|\tilde{\mathbf{I}}_i^{(t+1)}, \tilde{w}^{(t)})$ by the Langevin dynamics, which is initialized from the X_i generated from the prior distribution.

The interaction between the generator model and the MCMC can be illustrated by the following diagram (assuming that the generator is of enough capacity to approximate any distribution):

$$\begin{array}{rccc} \text{MCMC}: & P^{(t)} & \xrightarrow{\text{Markov transitions}} & P^{(t+1)} \\ & \Updownarrow & & \Updownarrow \\ \text{Generator}: & \tilde{w}^{(t)} & \xrightarrow{\text{Parameter updating}} & \tilde{w}^{(t+1)} \end{array} \tag{52}$$

In each iteration t, the generator provides a fresh new batch $\{\tilde{\mathbf{I}}_i^{(t)}, i = 1, ..., \tilde{n}\}$. We then run a finite number of MCMC transitions from $\{\tilde{\mathbf{I}}_i^{(t)}, i = 1, ..., \tilde{n}\}$ to obtain $\{\tilde{\mathbf{I}}_i^{(t+1)}, i = 1, ..., \tilde{n}\}$. After that, we let the generator model reconstruct $\{\tilde{\mathbf{I}}_i^{(t+1)}, i = 1, ..., \tilde{n}\}$, with essentially known X_i that generates $\tilde{\mathbf{I}}_i^{(t)}$, as illustrated by the diagrams in (50), in order for the generator to shift its density from $P^{(t)}$, which is the distribution of $\{\tilde{\mathbf{I}}_i^{(t)}\}$, to $P^{(t+1)}$, which is the distribution of $\{\tilde{\mathbf{I}}_i^{(t+1)}\}$.

In the above learning scheme, the deep FRAME model and the generator model cooperate with each other like a teacher and a student, where the deep FRAME model plays the role of the teacher, and the generator model plays the role of the student. It is as if the student writes up the initial draft of the paper. The teacher then revises it. After that, the teacher learns from

Figure 11: Generating texture patterns. For each category, the first image (224×224) is the training image, and the rest are 3 of the images generated by the cooperative learning algorithm.

Figure 12: Generating object patterns. For each object category, the first 3 images (64×64) are 3 of the training images, and the rest are 3 of the images generated by the cooperative learning algorithm.

the outside review, while the student learns from how the teacher revises the initial draft.

[88] provides a theoretical understanding of the cooperative learning algorithm. The learning of the deep FRAME model $p(\mathbf{I}; w)$ follows a modified version of the contrastive divergence method [38], where the generator $q(\tilde{\mathbf{I}}; \tilde{w})$ provides examples to initialize MCMC sampling of $p(\mathbf{I}; w)$. The update of the generator $q(\tilde{\mathbf{I}}; \tilde{w})$ seeks to approximate the Markov transition from $\tilde{\mathbf{I}}_i^{(t)}$ to $\tilde{\mathbf{I}}_i^{(t+1)}$, more specifically, $q(\tilde{\mathbf{I}}; \tilde{w})$ seeks to be the stationary distribution of this Markov transition, and the stationary distribution is nothing but $p(\mathbf{I}; w)$. If the generator has infinite learning capacity, then it will replicate the deep FRAME model perfectly. The analysis of the more realistic situation where there is discrepancy between the generator model and the deep FRAME model is much more complicated, which we shall study in our future work.

Figure 11 displays the results of learning texture patterns. Figure 12 displays the results of learning object patterns.

We then conduct an experiment on synthesizing images of categories from MIT places205 dataset [94]. We adopt a 4-layer network for $f(\mathbf{I}; w)$. The first layer has 64 5×5 filters with sub-sampling of 2 pixels, the second layers has 128 3×3 filters with sub-sampling of 2, the third layer has 256 3×3 filters with sub-sampling of 1, and the final layer is a fully connected layer with 100 channels as output. We continue to use the structure of the

Figure 13: Generating scene images (64 × 64). For each category, the left panel consists of randomly sampled training images. The right panel consists of randomly sampled synthetic images generated by the learned models. The categories are from MIT places205 dataset.

generator network of [69, 18]. We set the number of Langevin dynamics steps in each learning iteration to 10. For each category, we learn $f(\mathbf{I}; w)$ and $g(X; \tilde{w})$ from all the 10,000+ images in this category where we resize the images to 64×64. We run about 1,000 iterations. Figure 13 displays the results for two categories, where for each category, we show 144 randomly sampled observed images on the left, and 144 randomly sampled synthetic images generated by our method on the right.

7. Adversarial interpretation

The deep convolutional energy-based model (31) can be written as

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp[-U(\mathbf{I}; w)], \tag{53}$$

where the energy function $U(\mathbf{I}; w) = -f(\mathbf{I}; w) + \frac{1}{2\sigma^2}||\mathbf{I}||^2$. The update of w is based on $L'(w)$ which can be approximated by

$$\frac{\partial}{\partial w}\left[\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} U(\tilde{\mathbf{I}}_i; w) - \frac{1}{n}\sum_{i=1}^{n} U(\mathbf{I}_i; w)\right], \tag{54}$$

where $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthetic images that are generated by the Langevin dynamics. At the zero temperature limit, the Langevin dynamics becomes gradient descent:

$$\tilde{\mathbf{I}}_{\tau+1} = \tilde{\mathbf{I}}_\tau - \delta \frac{\partial}{\partial \tilde{\mathbf{I}}} U(\tilde{\mathbf{I}}_\tau; w). \tag{55}$$

Consider the value function

$$V(\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}; w) = \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} U(\tilde{\mathbf{I}}_i; w) - \frac{1}{n}\sum_{i=1}^{n} U(\mathbf{I}_i; w). \tag{56}$$

The updating of w is to increase V by shifting the low energy regions from the synthetic images $\{\tilde{\mathbf{I}}_i\}$ to the observed images $\{\mathbf{I}_i\}$, whereas the updating of $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ is to decrease V by moving the synthetic images towards the low energy regions. This is an adversarial interpretation of the learning and sampling algorithm [91]. It can also be considered as a generalization of the herding method [82] for the exponential family models to general energy-based models.

If we recruit a generator model $g(X; \tilde{w})$ as a sampler, then the energy-based model and the generator model play a minimax game with the value function

$$V(\tilde{w}; w) = \mathrm{E}_{\tilde{w}}[U(\tilde{\mathbf{I}}; w)] - \mathrm{E}_{\mathrm{data}}[U(\mathbf{I}; w)], \tag{57}$$

where $\mathrm{E}_{\tilde{w}}$ is the expectation with respect to the generator model with parameter $\tilde{w}$, and $\mathrm{E}_{\mathrm{data}}[U(\mathbf{I}; w)] = \frac{1}{n}\sum_{i=1}^{n} U(\mathbf{I}_i; w)$. This is related to [1].

8. Discussion

This paper reviews the sparse and deep generalizations of the FRAME model, and explains an auto-encoding structure and an adversarial interpretation.

Besides the FRAME model and its generalizations, there are other generative models, such as deep Boltzmann machines [40, 75, 54], auto-regressive models [67, 15, 16]. As to the generator network, it can also be learned by generative adversarial learning [69, 13, 6, 1], or variational auto-encoder [49, 71, 60], or the wake-sleep algorithm [39].

Recently an introspective learning method has been proposed by [51, 45], by generalizing Tu's original proposal in 2007 [79] to deep neural networks. This method learns a deep energy-based model by training a discriminative model. The discriminative model seeks to tell apart the synthesized examples generated by the current energy-based model from the real examples. The learned discriminative model can then be used to update the energy-based model so that it can generate new synthesized examples to pass the discriminative model. Repeating this process enables learning of both discriminative and generative models.

While the sparse FRAME model is interpretable in terms of symbolic sketch of the images, the deep FRAME model is not interpretable with its multiple layers of dense connections in linear filtering. Perhaps the non-interpretability of the deep ConvNets is a fact we have to live with, very much like we find peace with quantum mechanics with its unitary linear evolution of the wave function and non-linear probabilistic collapsing of the wave function at measurement, as long as it is mathematically consistent and it gives correct predictions. The dense connections may be doing some implicit form of Bayesian model averaging without explicitly inferring latent variables whose uncertainties may be too large to be worthy of explicit inference, especially at the lower layers. On the other hand, at the higher layers, sparse connections and symbolic representations, as well as grammatical understanding [98] and logical reasoning, may naturally emerge, as the uncertainties become smaller. It would be interesting to find out how such sparse and symbolic representations arise from dense continuous representations.

Another aspect of the ConvNet is that it blurs the boundary between representation and computation. While the nodes in the ConvNet may represent certain features or non-linear dimensions in the data, a ConvNet may also encode a computational algorithm. For example, in the cooperative learning algorithm, we may consider the generator model as encoding a

non-iterative sampling algorithm that reproduces the iterative MCMC sampling by accumulating and memorizing the effect of MCMC transitions. As another example, the variational auto-encoder method recruits an inference ConvNet that maps the image to the latent factors. This inference ConvNet actually encodes the computation of the posterior sampling of the latent factors. It appears that we not only learn the models with the ConvNet, we can also learn the computations in sampling and inference with the ConvNet.

A most surprising fact about the ConvNet is that even though the learning objective function is highly non-convex, the simple stochastic gradient descent algorithm works very well in practice. Because each iteration of the stochastic gradient only requires operating on a mini-batch, it can easily scale up to big data. Even though the stochastic gradient can only hope to get close to a local minima, this may actually be an advantage in the sense that the local modes and the randomness or noises may provide regularization to avoid over-fitting. Our current theoretical understanding of this issue is still rather limited, despite some recent progresses [61, 7, 70]. It is our hope that some mathematically minded readers of this paper may offer some theoretical insights into this issue.

Acknolwedgment

The work is supported by NSF DMS 1310391, DARPA SIMPLEX N66001-15-C-4035, ONR MURI N00014-16-1-2007, and DARPA ARO W911NF-16-1-0579.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974. MR0373208

[3] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984. MR0726392

[4] Hilton Bristow, Anders Eriksson, and Simon Lucey. Fast convolutional sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 391–398, 2013.

[5] 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. MR1639094

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2172–2180, 2016.

[7] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. *arXiv preprint arXiv:1412.0233*, 2014.

[8] Jifeng Dai, Yang Lu, and Ying Nian Wu. Generative modeling of convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2015. MR3553376

[9] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

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

[11] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977. MR0501537

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

[13] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1486–1494, 2015.

[14] Persi Diaconis and David Freedman. On the statistics of vision: the julesz conjecture. *Journal of Mathematical Psychology*, 24(2):112–138, 1981. MR0640207

[15] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[17] David Leigh Donoho. Sparse components of images and optimal atomic decompositions. *Constructive Approximation*, 17(3):353–382, 2001. MR1828917

[18] E Dosovitskiy, J T Springenberg, and T Brox. Learning to generate chairs with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] M Elad, M Aharon, and AM Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Transactions on Signal Processing*, 15(12):3736–3745, 2006. MR3018320

[20] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010. MR2677506

[21] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002. MR1951635

[22] 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. MR1473055

[23] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000. MR1790002

[24] Jerome H Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, pages 1–67, 1991. MR1091842

[25] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001. MR1873328

[26] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning multi-grid generative convnets by minimal contrastive divergence. *arXiv preprint arXiv:1709.08868*, 2017.

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

[28] Stuart Geman and Christine Graffigne. Markov random field image models and their applications to computer vision. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2, 1986.

[29] J Willard Gibbs. *Elementary principles in statistical mechanics*. Courier Corporation, 2014. MR0116523

[30] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011. MR2814492

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. MR3617773

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[33] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference*. Oxford University Press, 2007. MR2285439

[34] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *AAAI*, 2017.

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009. MR2722294

[36] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[37] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[38] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[39] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[40] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. MR2224485

[41] Yi Hong, Zhangzhang Si, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Unsupervised learning of compositional sparse code for natural image representation. *Quarterly of Applied Mathematics*, 72:373–406, 2013. MR3186243

[42] John Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, 1982.

[43] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982. MR0652033

[44] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005. MR2249836

[45] Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classification with convolutional nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[46] Bela Julesz. Visual pattern discrimination. *IRE transactions on Information Theory*, 8(2):84–92, 1962.

[47] Bela Julesz et al. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[48] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

[49] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[50] 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.

[51] Justin Lazarow, Long Jin, and Zhuowen Tu. Introspective neural networks for generative modeling. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[52] 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.

[53] Yann LeCun, Sumit Chopra, Rata Hadsell, Mare'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

[54] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning (ICML)*, pages 609–616, 2009.

[55] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008. MR2401592

[56] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015.

[57] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Learning FRAME models using CNN filters. In *AAAI*, 2016.

[58] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 1993. MR2690364

[59] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London B: Biological Sciences*, 204(1156):301–328, 1979.

[60] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning (ICML)*, 2014.

[61] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2924–2932, 2014.

[62] David Mumford and Agnès Desolneux. *Pattern theory: the stochastic analysis of real-world signals*. CRC Press, 2010. MR2723182

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

[64] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[65] Jiquan Ngiam, Zhenghao Chen, Pang Wei Koh, and Andrew Y Ng. Learning deep energy models. In *International Conference on Machine Learning (ICML)*, 2011.

[66] 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.

[67] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[68] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[69] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[70] Benjamin Recht, Moritz Hardt, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[71] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

[72] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[73] 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.

[74] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867, 2005.

[75] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *AISTATS*, 2009.

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[77] Kevin Swersky, Marc'Aurelio Ranzato, David Buchman, Benjamin Marlin, and Nando Freitas. On autoencoders and score matching for energy based models. In *International Conference on Machine Learning (ICML)*, pages 1201–1208, 2011.

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

[79] Zhuowen Tu, Learning Generative Models via Discriminative Approaches, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007

[80] Pascal Vincent. A connection between score matching and denoising autoencoders, 2010. *Neural computation*, 23(7):1661–1674, 2011.

[81] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008.

[82] Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning (ICML)*, pages 1121–1128, 2009.

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

[84] 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.

[85] Ying Nian Wu, Song-Chun Zhu, and Cheng-En Guo. From information scaling of natural images to regimes of statistical models. *Quarterly of Applied Mathematics*, 66:81–122, 2008.

[86] Ying Nian Wu, Song-Chun Zhu, and Xiuwen Liu. Equivalence of julesz ensembles and FRAME models. *International Journal of Computer Vision*, 38:247–265, 2000.

[87] 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.

[88] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *arXiv preprint arXiv:1609.09408*, 2016.

[89] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Inducing wavelets into random fields via generative boosting. *Journal of Applied and Computational Harmonic Analysis*, 41:4–25, 2016.

[90] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *International Conference on Machine Learning (ICML)*, 2016.

[91] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[92] 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.

[93] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2018–2025, 2011.

[94] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–495, 2014.

[95] Song-Chun Zhu. Statistical modeling and conceptualization of visual patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):691–712, 2003.

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

[97] Song-Chun Zhu and David Mumford. Grade: Gibbs reaction and diffusion equations. In *IEEE International Conference on Computer Vision (ICCV)*, pages 847–854, 1998.

[98] Song-Chun Zhu and David Mumford. *A stochastic grammar of images*. Now Publishers Inc, 2007.

[99] 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.

Ying Nian Wu
Department of Statistics
University of California
Los Angeles
E-mail address: ywu@stat.ucla.edu

Jianwen Xie
Hikvision Research Institute
Santa Clara, CA
USA
E-mail address: jianwen@ucla.edu

Yang Lu
Amazon RSML (Retail System Machine Learning) Group
E-mail address: yanglv@ucla.edu

Song-Chun Zhu
Department of Statistics
University of California
Los Angeles
E-mail address: sczhu@stat.ucla.edu

Received June 26, 2017