Statistics and Its Interface Volume 11 (2018) 515–529

Exploring generative perspective of convolutional neural networks by learning random field models[*]

Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu[†]

This paper studies the convolutional neural network (ConvNet or CNN) from a statistical modeling perspective. The ConvNet has proven to be a very successful discriminative learning machine. In this paper, we explore the generative perspective of the ConvNet. We propose to learn Markov random field models called FRAME (Filters, Random field, And Maximum Entropy) models using the highly sophisticated filters pre-learned by the ConvNet on the big ImageNet dataset. We show that the learned models can generate realistic and rich object and texture patterns in natural scenes. We explain that each learned model corresponds to a new ConvNet unit at the layer above the layer of filters employed by the model. We further show that it is possible to learn a generative ConvNet model with a new layer of multiple filters, and the learning algorithm admits an EM interpretation with binary latent variables.

AMS 2000 subject classifications: Generative models, Langevin dynamics, Markov random fields, Energy-based models.

1. INTRODUCTION

In this section, we present the recent history of the convolutional neural network and explain its statistical groundings in generalized linear model and Markov random field. We also explain the objective of our work and the statistical generative models that we shall present.

1.1 Recent history: ConvNet met ImageNet

The breakthrough made by the convolutional neural network (ConvNet or CNN) [19, 21] on the ImageNet dataset [6] in 2012 was a watershed event in machine learning that has transformed several fields in artificial intelligence, such as computer vision, speech recognition, natural language processing, etc., as well as related industries. The neural networks in general and the ConvNets in particular were developed in the 1980s and 1990s respectively, but they had to wait for the much improved computing power brought by GPUs and much bigger datasets such as ImageNet to fully realize their potential.

The ImageNet dataset, first released in 2009, is a collection of more than 15 million natural images organized into roughly 22,000 categories. The categories are taken from the visually meaningful nouns in the WordNet, a comprehensive database of English words. The images were collected by querying the categories on the internet search engines such as Google, and were manually examined by crowdsourcing workers from Amazon's Mechanical Turk. Starting from 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [29] has been held annually. In its image classification task, which is to assign each image to an object category, there are roughly 1.2 million training images, 50,000 validation images, and 100,000 testing images, from a 1,000 category subset of the original ImageNet dataset.

In ILSVRC 2012, the ConvNet [19] emerged as the runaway winner of the image classification competition. The winning network, now commonly dubbed as the AlexNet (after the first name of the first author), has 60 million parameters and 650,000 hidden nodes. It consists of 5 convolutional layers (some of them are followed by sub-sampling and max-pooling layers) and 3 fully-connected layers. Since then, ConvNets as well as other neural networks under the banner of "deep learning" [1] have become widely adopted for many tasks in artificial intelligence, such as those in computer vision, speech recognition, natural language processing, etc., and have achieved state of the art performances, sometimes super-human performances, on these tasks.

One interesting phenomenon is that the features or the non-linear filters learned by the ConvNets on the ImageNet dataset have been shown to be highly effective for many computer vision tasks. They usually outperform existing handcrafted features by big margins. Clearly the ConvNets have learned meaningful features that characterize the natural images such as those in ImageNet.

Despite its tremendous successes, the practice of ConvNet is still very empirical. The ConvNet is designed empirically. The learning is based on gradient descent algorithm on a highly non-convex multi-modal objective function.

1.2 Objective: exploring generative perspective of ConvNet

This paper is a case study of ConvNet trained by ImageNet from a statistical modeling perspective. A ConvNet

*The authors gratefully acknowledge NSF DMS 1310391, ONR MURI N00014-10-1-0933, DARPA SIMPLEX N66001-15-C-4035, and DARPA MSEE FA 8650-11-1-7149.

†Corresponding author.

is a discriminative or predictive machine. It learns to predict the object category or class label of the input image. More specifically, it learns a highly non-linear mapping (or function, or classifier) where the input is the image and the output is the category. Such a discriminative machine tells us *how* to recognize a pattern from an image, such as how to recognize a hummingbird or how to tell a hummingbird apart from, say, a seagull. In contrast, the generative direction tells us *what* a hummingbird looks like by producing sample images of hummingbirds. While the discriminative direction is from image to category, the generative direction is the opposite. The discriminative direction is operational, and the generative direction is representational. One may intuitively consider the generative direction as a matter of imagination, a gift that was obviously possessed by a human brain. See Fig. 4 for an illustration of the two directions.

While ConvNet has proven to be a powerful discriminative machine, researchers have recently become increasingly interested in the generative perspective of ConvNet. An interesting example is the Google deep dream (http://deepdreamgenerator.com/), which generates interestingly vivid images.

In this paper, we explore the generative perspective of ConvNet more formally by defining formal statistical generative models based on ConvNet features pre-trained by ImageNet, and learning these models by generating images from the models. These models are defined on images, so they are random field models. Adopting the metaphor of Google deep dream, we let the random field models dream by generating images from the models. But unlike the Google deep dream, we learn the models from real images by matching the dreamed-up images to real images, i.e., by making the dreams come true.

From a statistician's perspective, generative models are more natural representations of knowledge because they tell us what the patterns look like. It is more interesting to find statistical models to explain the observed images than to predict the class labels of the images, especially because the images of natural scenes contain such a bewildering variety of patterns [32]. In this paper, we shall show that our random field models based on ConvNet features can generate realistic and rich object and texture patterns in natural scenes.

We shall first learn generative models from images of aligned objects. Fig. 1 shows 2 experiments. In each experiment, the first row displays 4 of the training images. The second row displays 4 of the images generated by the learned model. The training images are collected from the internet. For each category, the number of training images is around 10. We shall also learn generative models from texture images. Fig. 2 shows two experiments. Each experiment is displayed by 3 images, where the first image is the single training image, and the remaining 2 images are generated by the learned model. In addition, we shall also learn generative models from images where the patterns are not aligned.

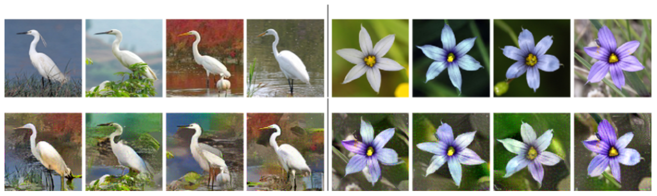


Figure 1. Generating object patterns. For each category, the first row displays 4 of the training images, and the second row displays 4 of the images generated by the learned random field model.

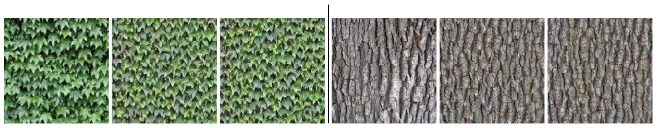


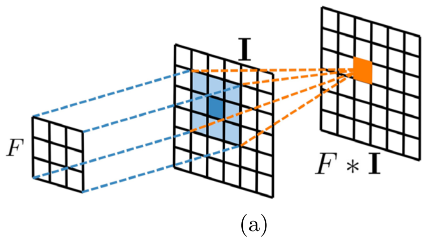
Figure 2. Generating texture patterns. For each category, the first image is the single training image, and the next 2 are images generated by the learned random field model.

The generative models help us understand knowledge representation in ConvNet. They help us confirm that, collectively, the ConvNet features pre-trained on ImageNet are very expressive in describing the natural images. More importantly, the generative models may eventually enable us to learn from natural images or other types of data from scratch in an unsupervised manner without requiring the class labels or annotations of the input data. The acquisition of the class labels or annotations can be time consuming and expensive. Guided by the likelihoods of the generative models, the features can be learned by finding the best explanations of the input data instead of finding the best predictions of the output labels.

In this case study paper we shall restrict ourselves to learning generative models from small sets of training images using the existing ConvNet features pre-trained on ImageNet of labeled images. This is like acquiring new knowledge from recent experiences based on the existing knowledge accumulated from all the past experiences.

1.3 Statistical groundings on GLM and MRF

For statisticians, a feedforward neural network can be viewed as a generalization of the generalized linear model (GLM). A GLM, such as logistic regression, is characterized by a composition of a linear combination or weighted sum of the predictor variables and a one-dimensional non-linear link function. A feedforward neural network or multi-layer perceptron is a recursion of such a compositional scheme, where each predictor variable itself is defined by a non-linear link function of a linear combination or weighted sum of predictor variables at the lower layer. The predictor variables at the bottom layer are the raw input variables. In the terminology of neural networks, each predictor variable at each layer is called a unit, a node, a feature, or a filter. The neural



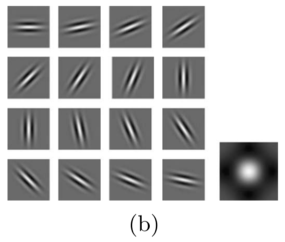


Figure 3. (a) Filtering or convolution: applying a filter (3×3) on an image (6×6) to get a filtered image (6×6, with proper boundary handling) or feature map. Each pixel of the filtered image is computed by the weighted sum of the 3×3 pixels of the input image centered at this pixel. (b) Gabor filters (wavelets) at different orientations, and Difference of Gaussians (DoG) filter (the rightmost one). The Gabor filters are sine and cosine waves multiplied by elongated Gaussian functions. The DoG filter is the difference between two isotropic Gaussian functions of different scales. The filters can appear at different locations and scales.

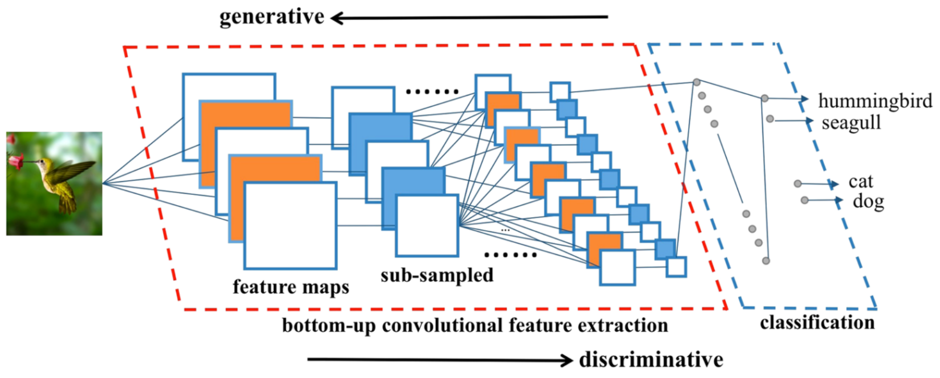


Figure 4. Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image. This illustration is adapted from [21].

network is able to learn multiple layers of features instead of handcrafting them based on the domain knowledge.

The ConvNet is a variation of feedforward neural network and is often deployed to analyze signals such as image data, where the linear combinations or weighted sums are computed locally around every pixel in a translation invariant or "convolutional" manner. The weights of a local weighted summation define a filter, and a local weighted sum is called a filter response. See Fig. 3 (a) for an illustration of a linear filter. A filter performs the same local summation operation around each pixel, thus producing a filter response or a feature at each pixel. The filter responses or features extracted by the same filter form a filtered image or feature map. At each layer of ConvNet, there can be many filters, extracting many maps of features. Each feature will then go through a non-linear transformation, so the non-linear transformation is applied element-wise on the feature maps. The feature maps may also go through sub-sampling, e.g., we may keep a feature every 2 pixels in both directions, so that the sizes of the feature maps are reduced after sub-sampling. See Fig. 4 for an illustration of a ConvNet. Eventually, the features at the top layer are used for predicting the category of the input image using multinomial logistic regression.

Fig. 3 (b) displays two types of filters, namely, the Gabor filters and the Difference of Gaussian (DoG) filters, that are commonly used in image processing. The Gabor filters and DoG filters are handcrafted, guided by the neuroscience observations on the primary visual cortex. In ConvNet, however, such filters are to be learned from the training data such as ImageNet. It is interesting that the linear filters at the bottom layer of the ConvNet trained on ImageNet resemble the Gabor filters and DoG filters [19].

While the GLM can be considered the statistical grounding of ConvNet, the Markov random field (MRF) or equivalently the Gibbs distribution [2, 11] can be considered the statistical grounding of the generative perspective of ConvNet that we shall explore in this paper. An MRF or a Gibbs distribution is a probability distribution defined on image. The log probability density or the energy function of a Gibbs distribution involves sum of functions defined on cliques, which are sets of pixels that are neighbors of each other. To connect to ConvNet, the clique functions can be defined by the ConvNet filters, and they can be learned from the data by maximum likelihood.

1.4 Our work: FRAME models using ConvNet filters

We propose to explore the generative perspective of ConvNet by learning the FRAME (Filters, Random field, And Maximum Entropy) models [44, 37] using the highly sophicated filters pre-learned by ConvNet on the ImageNet dataset. A FRAME model is a Markov random field model that defines a probability distribution on the image space. It is also an exponential family model whose log probability density consists of compositions of linear filtering and element-wise non-linear transformation. The model is generative in the sense that images can be generated from the probability distribution defined by the model. The probability distribution is the maximum entropy distribution that reproduces the statistical properties of filter responses in the observed images. Being of the maximum entropy, the distribution is the most random distribution that matches the observed statistical properties of filter responses, so that images sampled from this distribution can be considered typical images that share the statistical properties of the observed images.

There are two versions of FRAME models in the literature. The original version is a stationary model developed for modeling texture patterns [44], such as those in Fig. 2. The more recent version is a non-stationary extension designed to represent object patterns [37], such as those in Fig. 1. Both versions of the FRAME models can be sparsified by selecting a subset of filters from a given dictionary.

The filters used in the FRAME models are the oriented and elongated Gabor filters at different scales, as well as the isotropic Difference of Gaussian (DoG) filters of different sizes, see Fig. 3 (b). These are linear filters that capture simple local image features such as edges and blobs. With the emergence of the more powerful non-linear filters learned by ConvNet at various convolutional layers from ImageNet, it is only natural to replace the linear filters in the original FRAME models by the ConvNet filters in the hope of learning more expressive models.

We use the Langevin dynamics [23] to sample from the probability distribution defined by the model. Such a dynamics was first applied to the FRAME model by [41], and the gradient descent part of the dynamics was interpreted as the Gibbs Reaction And Diffusion Equations (GRADE). When applied to the FRAME model with ConvNet filters, the dynamics can be viewed as a recurrent generative form of the model, where the reactions and diffusions are governed by the ConvNet filters of positive and negative weights respectively.

Incorporating ConvNet filters into the FRAME model leads to a seamless meshing between the FRAME model and the ConvNet model. The original FRAME model has an energy function that consists of a layer of linear filtering followed by a layer of element-wise non-linear transformations. It is natural to follow the deep learning philosophy to expand them into alternative layers of linear filtering and non-linear transformations to have a deep FRAME model that directly corresponds to a ConvNet. More importantly, the learned FRAME model using ConvNet filters corresponds to a new ConvNet unit at the layer directly above the layer of ConvNet filters employed by the FRAME model. In particular, the non-stationary FRAME that generates images like those in Fig. 1 becomes a single ConvNet node at a specific position where the object appears, whereas the stationary FRAME that generates images like those in Fig. 2 becomes a special type of convolutional unit. Therefore, the learned FRAME model can be viewed as a generative version of a ConvNet unit.

In addition to learning a single ConvNet unit, we can also learn a generative model that involves a new layer of multiple convolutional units from non-aligned images, so that each convolutional unit represents one type of local pattern. We call the resulting model the generative ConvNet model. It is a product of experts model [14], where each expert models a mixture of presence and absence of a local pattern. The rectified linear unit, which is the non-linear link function commonly adopted in modern ConvNet [19], can be justified as an approximation to the log-likelihood function of this mixture model. The learning algorithm admits an interpretation in terms of the EM algorithm [5] with a hard-decision E-step that detects the local patterns modeled by the convolutional units.

By exploring the generative perspective of ConvNet, this paper establishes the conceptual correspondence between the generative FRAME model and the discriminative ConvNet, thus providing a formal generative foundation for ConvNet. As mentioned above, such a foundation is much needed because it may eventually lead to unsupervised learning of ConvNet in a generative fashion without the need for image labeling.

1.5 Related work on generative ConvNet

Recently there have been many interesting papers on visualizing ConvNet nodes, such as deconvolutional networks [40], score maximization [30], and the recent artful work of Google deep dream (http://deepdreamgenerator.com/) and painting style [10]. Our work is different from these previous methods in that we learn rigorously defined generative models from training images, and the learned models correspond to new ConvNet units. This work is a continuation of the recent work on generative ConvNet [3].

There have also been recent papers on generative models based on supervised image generation [8], variational auto-encoders [15, 18, 28, 26, 20, 12], and adversarial networks [7]. Each of these papers learns a top-down multi-layer model for image generation, but the parameters of the top-down generation model are completely separated from the parameters of the bottom-up recognition model. Our work seeks to learn a generative model based on the knowledge learned by the bottom-up recognition model, i.e., the image generation model and the image recognition model share the same set of parameters.

The paper is an expanded version of a previous conference paper [25]. Some of its contents have been reviewed by [35], which also reviews related recent developments.

2. TECHNICAL BACKGROUND ON CONVNET

This section reviews the technical background of ConvNet. We first explain that a linear filter is a local linear model. We then explain that a ConvNet is a recursive composition of generalized linear models.

2.1 Filters: local linear models

To fix notation, 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.

A linear filter is just a local weighted sum of image intensities around each pixel. 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, such as padding zeros for pixels outside the 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 or coefficients $(w_{k,x}, x \in \mathcal{S})$ define the filter F_k, and $\mathcal{S}$ is the localized support of the filter centered at the origin. See Fig. 3 (a) 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 said to be a convolution operation, where $(w_{k,x}, x \in \mathcal{S})$ form the kernel function of the convolution. Here "convolutional" means that the operations are shift-invariant or translation-invariant.

Compared to the linear model in regression and GLM, the image intensities play the role of input variables, and the weight parameters $w_{k,x}$ play the role of coefficients. These parameters are to be learned from the data.

2.2 ConvNet: GLMs on top of GLMs

A ConvNet is a composition of multiple layers of linear filtering and element-wise non-linear transformations, as expressed by the following recursive formula:

$$\begin{aligned}[F_j^{(l)} * \mathbf{I}](y) \\ = h\left(\sum_{k=1}^{N_{l-1}} \sum_{x \in \mathcal{S}_l} w_{k,x}^{(l,j)} [F_k^{(l-1)} * \mathbf{I}](y + x) + b_{l,j}\right),\end{aligned} \tag{2}$$

where $l \in \{1, 2, ..., L\}$ indexes the layer. $\{F_j^{(l)}, j = 1, ..., N_l\}$ 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 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 define the image $\mathbf{I}$ to be the feature map at the 0-th layer. 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 $h()$, sometimes called rectification function.

Compared to GLM, the weight parameters $w_{k,x}^{(l,j)}$ and the bias term $b_{l,j}$ correspond to the coefficients and the intercept of a GLM, and the features at layer $l - 1$ are the predictor variables for computing the features at layer l. The non-linear transform $h()$ plays the role of the link function of a GLM. Just as in GLM, the link function $h()$ is specified, and the weight and bias parameters are to be learned from the data. Due to the recursive nature of equation (2), a ConvNet can be considered GLMs on top of GLMs.

From the perspective of filters, $\{F_j^{(l)}\}$ are non-linear filters because we incorporate $h()$ in the computation of the filter responses. We call $F_j^{(l)} * \mathbf{I}$ the filtered image or the feature map of filter j at layer l. There are a total N_l feature maps in layer l, and $j = 1, ..., N_l$. In Fig. 4, the feature maps are illustrated by the square shapes. Each $[F_j^{(l)} * \mathbf{I}](x)$ is called a feature extracted by a node or a unit at layer l.

The filtering operations are often followed by sub-sampling and local-max pooling (e.g., $\mathbf{I}(x_1, x_2) \leftarrow \max_{(\delta_1,\delta_2) \in \{0,1\}^2} \mathbf{I}(2x_1 + \delta_1, 2x_2 + \delta_2)$). See Fig. 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. Beyond that point, the network becomes fully connected between adjacent layers. These layers are called fully connected layers, and the layers below are called convolutional layers.

The features at the top layer are then used for classification (e.g., does the image contain a hummingbird or a seagull or a dog) via multinomial logistic regression. Specifically, let the top layer filter responses or features be $\{F_k^{(L)} * \mathbf{I}, k = 1, ..., N_L\}$. Let $c \in \{0, 1, ..., C\}$ be the category of image $\mathbf{I}$, then the score is

$$f_c(\mathbf{I}; w) = \sum_k w_{c,k} [F_k^{(L)} * \mathbf{I}] + b_{c,k}, \tag{3}$$

where $w_{c,k}$ and $b_{c,k}$ are the weights (coefficients) and bias (intercept) for computing the score of category c, and the parameter w includes the category-specific $w_{c,k}$ and $b_{c,k}$, as well as the weight and bias parameters at all the layers below, which are shared by all the categories. The conditional probability of the category c given the image $\mathbf{I}$ is

$$p(c|\mathbf{I}, w) = \frac{\exp(f_c(\mathbf{I}; w))}{\sum_c \exp(f_c(\mathbf{I}; w))}. \tag{4}$$

For identifiability, we may choose a base category, e.g., background, with $c = 0$, and define $f_0(\mathbf{I}) = 0$.

The estimation of the weight and bias parameters can be accomplished by gradient ascent on the log-likelihood, i.e., $L(w) = \sum_{(\mathbf{I},c)} \log p(c|\mathbf{I}, w)$ over all the labeled examples $\{(\mathbf{I}, c)\}$. For big data, we can divide the data into mini-batches, so that at each step, we run gradient ascent based on the log-likelihood of a randomly sampled mini-batch. The gradient can be calculated by back-propagation, which is an application of the chain rule on the recursive composite function $L(w)$. The bottom layer filters of the ConvNet learned from the ImageNet data resemble the Gabor and DoG filters in Fig. 3 (b) [19].

3. FRAME MODELS BASED ON LINEAR FILTERS

This section reviews the background on the FRAME models based on linear filters. The FRAME models are a class of Markov random field models or Gibbs distributions, where the energy functions consist of non-linear transformations of linear filter responses.

3.1 Stationary FRAME

Again, let $\mathbf{I}$ be an image defined on a square (or rectangular) domain $\mathcal{D}$. Let $\{F_k, k = 1, ..., K\}$ be a bank of linear filters, such as elongate and oriented Gabor filters at different scales, as well as isotropic Difference of Gaussian (DoG) filters of different sizes. Some examples of the filters are shown in Fig. 3 (b). Let $F_k * \mathbf{I}$ be the filtered image or feature map, and $[F_k * \mathbf{I}](x)$ be the filter response or feature at position x (again x is a two-dimensional coordinate).

The original FRAME model [44] for texture patterns, such as those in Fig. 2, is a stationary or spatially homogeneous Markov random field or Gibbs distribution of the following form:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} \lambda_k\left([F_k * \mathbf{I}](x)\right)\right], \tag{5}$$

where $\lambda_k()$ is a nonlinear function to be estimated from the training images, $\lambda = (\lambda_k(), k = 1, ..., K)$, and $Z(\lambda)$ is the normalizing constant to make $p(\mathbf{I}; \lambda)$ integrate to 1. In the original paper of [44], each $\lambda_k()$ is discretized and estimated as a step function, i.e., $\lambda_k(r) = \sum_{b=1}^{B} w_{k,b} h_b(r)$, where $b \in \{1, ..., B\}$ indexes the equally spaced bins of discretization, and $h_b(r) = 1$ if r is in bin b, and 0 otherwise, i.e., $h() = (h_b(), b = 1, ..., B)$ is a 1-hot indicator vector, and $\sum_x h([F_k * \mathbf{I}](x))$ is the marginal histogram of the filter map $F_k * \mathbf{I}$. The spatially pooled marginal histograms are the sufficient statistics of model (5).

Model (5) is stationary because the function $\lambda_k()$ does not depend on position x. In model (5), the energy function $U(\mathbf{I}; \lambda) = -\sum_k \sum_x \lambda_k([F_k * \mathbf{I}](x))$ involves a layer of linear filtering by $\{F_k\}$, followed by a layer of element-wise non-linear transformation by $\{\lambda_k()\}$. Repeating this pattern recursively (while also adding local max pooling and sub-sampling) will lead to a generative version of ConvNet.

3.2 Non-stationary FRAME

The non-stationary or spatially inhomogeneous FRAME model for object patterns [37], such as those in Fig. 1, is of the following form:

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

where the function $\lambda_{k,x}()$ depends on position x, and $\lambda = (\lambda_{k,x}(), \forall k, x)$. Again $Z(\lambda)$ is the normalizing constant. The model is non-stationary because $\lambda_{k,x}()$ depends on position x. It is impractical to estimate $\lambda_{k,x}()$ as a step function at each x, so $\lambda_{k,x}()$ is parametrized as a one-parameter function

$$\lambda_{k,x}(r) = w_{k,x} h(r), \tag{7}$$

where $h()$ is a pre-specified rectification function, and $w = (w_{k,x}, \forall k, x)$ are the unknown parameters to be estimated. In the paper of [37], they use $h(r) = |r|$ for full wave rectification. One can also use rectified linear unit $h(r) = \max(0, r)$ [19] for half wave rectification, which can be considered an elaborate two-bin discretization. $q(\mathbf{I})$ is a reference distribution, such as the Gaussian white noise model

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

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

In the original FRAME model (5), $q(\mathbf{I})$ is assumed to be a uniform measure. In model (6), we can also absorb $q(\mathbf{I})$, in particular, the $\frac{1}{2\sigma^2}||\mathbf{I}||^2$ term, into the energy function, so that the model is again defined relative to a uniform measure as in the original FRAME model (5). We make $q(\mathbf{I})$ explicit here because we shall specify the parameter σ^2 instead of learning it, and use $q(\mathbf{I})$ as the null model for the background. In models (6) and (7), $(w_{k,x}, \forall x, k)$ can be considered a second-layer linear filter on top of the first layer filters $\{F_k\}$ rectified by $h()$.

Both models (5) and (6) can be sparsified. Model (5) can be sparsified by selecting a small set of filters F_k using the filter pursuit procedure [44]. Model (6) can be sparsified by selecting a small number of filters F_k and positions x, so that only a small number of $w_{k,x}$ are non-zero. The sparsification can be achieved by a shared matching pursuit method [37] or a generative boosting method [38].

4. FRAME MODELS BASED ON CONVNET FILTERS

Instead of using linear filters, we can use the filters at various convolutional layers of a pre-learned ConvNet. We call such a model the deep FRAME model. Suppose there exists a bank of filters $\{F_k, k = 1, ..., K\}$ (e.g., $K = 512$) at a certain convolutional layer of a pre-learned ConvNet. For an image $\mathbf{I}$ defined on the square image domain $\mathcal{D}$, let $F_k * \mathbf{I}$ be the feature map of filter F_k, and let $[F_k * \mathbf{I}](x)$ be the filter response of $\mathbf{I}$ to F_k at position x (again x is a two-dimensional coordinate). We assume that $[F_k * \mathbf{I}](x)$ is the response obtained after applying the rectified linear transformation $h(r) = \max(0, r)$. $[F_k * \mathbf{I}](x)$ is defined recursively according to equation (2) in Section 2. For notational simplicity, we make the index of the layer, l, implicit.

Then the non-stationary deep FRAME model becomes

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K}\sum_{x\in\mathcal{D}} w_{k,x}[F_k * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{9}$$

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. $Z(w)$ is the normalizing constant. Model (9) shares the same form as model (6) with linear filters, except that the rectification function $h()$ in model (6) is already absorbed in the ConvNet filers $\{F_k\}$ in model (9) with $h(r) = \max(0, r)$. We shall use model (9) to generate object patterns such as those in Fig. 1.

The stationary FRAME model is of the following form:

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K}\sum_{x\in\mathcal{D}} w_k[F_k * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{10}$$

which is almost the same as model (9) except that w_k is the same across x. $w = (w_k, \forall k)$. We shall use model (10) to generate texture patterns such as those in Fig. 2.

Again, both models (9) and (10) can be sparsified, either by forward selection such as filter pursuit [44] or generative boosting [38], or by backward elimination.

5. LEARNING AND SAMPLING ALGORITHMS

This section presents the algorithms for learning and sampling from the FRAME models. Intuitively, the sampling algorithm is to let the model dream, and the learning algorithm is to update the model parameters by making the dreams come true. The sampling algorithm is an inner loop of the learning algorithm. That is, the model learns to dream and dreams to learn.

5.1 Learning algorithm

The basic learning algorithm for object model 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, where n is the total number of training images. In the basic learning algorithm, the weight parameters w can be estimated by maximizing the log-likelihood function

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} \log p(\mathbf{I}_i; w), \tag{11}$$

where $p(\mathbf{I}; w)$ is defined by (9). $L(w)$ is a concave function. The first derivatives of $L(w)$ are

$$\frac{\partial L(w)}{\partial w_{k,x}} = \frac{1}{n}\sum_{i=1}^{n}[F_k * \mathbf{I}_i](x) - \mathrm{E}_w\left([F_k * \mathbf{I}](x)\right), \tag{12}$$

where E_w denotes the expectation with respect to $p(\mathbf{I}; w)$. The expectation can be approximated by Monte Carlo integration. The second derivative of $L(w)$ is the variance-covariance matrix of $([F_k * \mathbf{I}](x), \forall k, x)$. w can be computed by a stochastic gradient ascent algorithm [39]:

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma\left[\frac{1}{n}\sum_{i=1}^{n}[F_k * \mathbf{I}_i](x) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}[F_k * \tilde{\mathbf{I}}_i](x)\right], \tag{13}$$

for every $k \in \{1, ..., K\}$ and $x \in \mathcal{D}$, where γ is the learning rate, and $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$ are the synthesized images sampled from $p(\mathbf{I}; w^{(t)})$ using MCMC. $\tilde{n}$ is the total number of independent parallel Markov chains that sample from $p(\mathbf{I}; w^{(t)})$. The learning rate γ can be made inversely proportional to the observed variance of $\{[F_k * \mathbf{I}_i](x), \forall i\}$, as well as being inversely proportional to the iteration t as in stochastic approximation.

For learning stationary FRAME (10), usually $n = 1$, i.e., we observe one texture image, and we update the parameters by

$$w_k^{(t+1)} = w_k^{(t)} + \frac{\gamma}{|\mathcal{D}|}\left[\frac{1}{n}\sum_{i=1}^{n}\sum_{x\in\mathcal{D}}[F_k * \mathbf{I}_i](x) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\sum_{x\in\mathcal{D}}[F_k * \tilde{\mathbf{I}}_i](x)\right], \tag{14}$$

for every $k \in \{1, ..., K\}$, where there is a spatial pooling across positions $x \in \mathcal{D}$.

5.2 Sampling algorithm

In order to sample from $p(\mathbf{I}; w)$ in (9), we adopt the Langevin dynamics. Writing the energy function

$$U(\mathbf{I}, w) = -\sum_{k=1}^{K}\sum_{x\in\mathcal{D}} w_{k,x}[F_k * \mathbf{I}](x) + \frac{1}{2\sigma^2}||\mathbf{I}||^2. \tag{15}$$

The Langevin dynamics iterates

$$\mathbf{I}_{\tau+1} = \mathbf{I}_\tau - \frac{\epsilon^2}{2} U'(\mathbf{I}_\tau, w) + \epsilon Z_\tau, \quad (16)$$

where $U'(\mathbf{I}, w) = \partial U(\mathbf{I}, w)/\partial \mathbf{I}$. This gradient can be computed by back-propagation. In (16), ϵ is a small step size, and $Z_\tau \sim \mathrm{N}(0, \mathbf{1})$, independently across τ, where the bold font $\mathbf{1}$ is the identify matrix, i.e., Z_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 is an inner loop within the learning process. Between every two consecutive updates of w in the learning process, we run a finite number of iterations of the Langevin dynamics starting from the images generated by the previous iteration of the learning algorithm, a scheme called "warm start" in the literature. The Langevin equation was also adopted by [41], who called the corresponding gradient descent algorithm the Gibbs reaction and diffusion equations (GRADE).

Algorithm 1 describes the details of the learning and sampling algorithms for the non-stationary model (9). The learning and sampling algorithms for the stationary model (10) only involve minor modifications of Algorithm 1. Algorithm 1 embodies the principle of "analysis by synthesis," i.e., we synthesize images by sampling from the current model, and then update the model parameters based on the difference between the synthesized images and the observed images.

Algorithm 1 Learning and sampling algorithms

```
Require:
    (1) training images $\{\mathbf{I}_i, i = 1, ..., n\}$
    (2) a filter bank $\{F_k, k = 1, ..., K\}$
    (3) number of synthesized images $\tilde{n}$
    (4) number of Langevin steps $L$
    (5) number of learning iterations $T$
Ensure:
    (1) estimated parameters $w = (w_{k,x}, \forall k, x)$
    (2) synthesized images $\{\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}\}$

1: Calculate observed statistics:
   $H^{\text{obs}}_{k,x} \leftarrow \frac{1}{n}\sum_{i=1}^{n}[F_k * \mathbf{I}_i](x), \forall k, x.$
2: Let $t \leftarrow 0$, initialize $w^{(0)}_{k,x} \leftarrow 0, \forall k, x.$
3: Initialize $\tilde{\mathbf{I}}_i \leftarrow 0$, for $i = 1, ..., \tilde{n}$.
4: repeat
5:     For each $i$, run $L$ steps of Langevin dynamics to update
       $\tilde{\mathbf{I}}_i$, i.e., starting from the current $\tilde{\mathbf{I}}_i$, each step updates
       $\tilde{\mathbf{I}}_i \leftarrow \tilde{\mathbf{I}}_i - \frac{\epsilon^2}{2}U'(\tilde{\mathbf{I}}_i, w^{(t)}) + \epsilon Z$, where $Z \sim \mathrm{N}(0, \mathbf{1})$.
6:     Calculate synthesized statistics:
       $H^{\text{syn}}_{k,x} \leftarrow \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}[F_k * \tilde{\mathbf{I}}_i](x), \forall k, x.$
7:     Update $w^{(t+1)}_{k,x} \leftarrow w^{(t)}_{k,x} + \gamma(H^{\text{obs}}_{k,x} - H^{\text{syn}}_{k,x}), \forall k, x.$
8:     Let $t \leftarrow t + 1$
9: until $t = T$
```

From the MCMC perspective, Algorithm 1 runs non-stationary parallel Markov chains that sample from a Gibbs distribution with a changing energy landscape, like in simulated annealing or tempering [23, 22]. This may help the chains to avoid the trapping of local modes. We can also use "cold start" scheme by initializing Langevin dynamics from white noise images in each learning iteration and allowing the dynamics enough time to relax.

6. IMAGE GENERATION EXPERIMENTS

In our experiments, we use the filters of the ConvNet learned by the VGG group [31] on the ImageNet dataset, and we use the Matlab code of MatConvNet [33].

Experiment 1: generating object patterns. We learn the non-stationary FRAME model (9) from images of aligned objects. The images are collected from the internet. For each category, the number of training images is around 10. We use $\tilde{n} = 16$ parallel chains for Langevin sampling. The number of Langevin iterations between every two consecutive updates of the parameters is $L = 100$. Fig. 5 shows some experiments using filters from the 3rd convolutional layer of VGG ConvNet. For each experiment, the first row displays 4 of the training images, and the second row displays 4 of the synthesized images generated by Algorithm 1.

Experiment 2: generating texture patterns. We learn the stationary FRAME model (10) from images of textures. Fig. 6 shows some experiments. Each experiment is displayed in one row, where the first image is the training

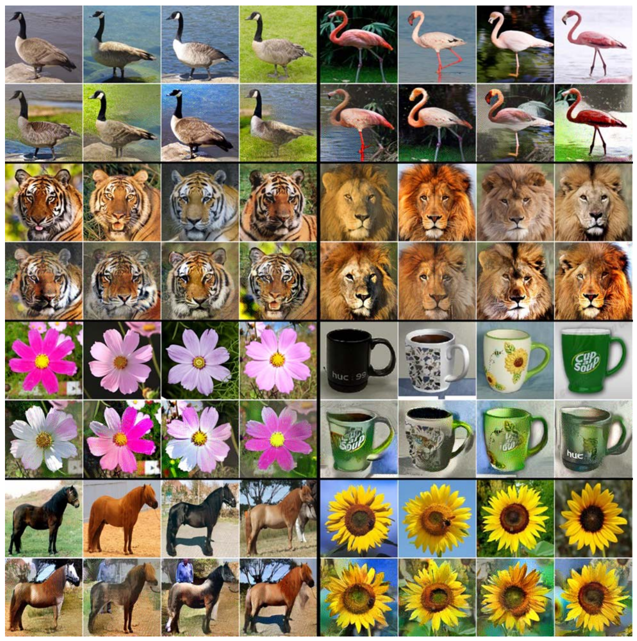


Figure 5. Generating object patterns. For each category, the first row displays 4 of the training images, and the second row displays generated images.

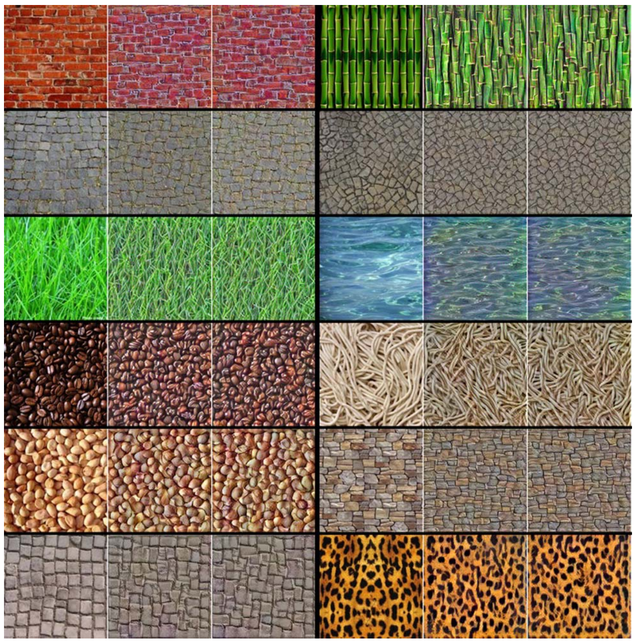


Figure 6. Generating texture patterns. For each category, the first image is the training image, and the next 2 images are generated images.

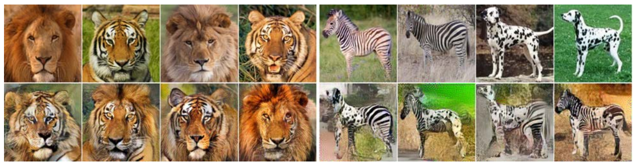


Figure 7. Generating hybrid object patterns. For each experiment, the first row displays 4 of the training images, and the second row displays generated images.

image, and the other 2 images are generated by the learning algorithm.

Experiment 3: generating hybrid patterns. We learn models (9) and (10) from images of mixed categories, and generate hybrid patterns. Figs 7 and 8 display a few examples. The non-stationary model re-mixes local image patterns from different images seamlessly, while the stationary model re-mixes and re-shuffles local image patterns seamlessly.

We also learn models (9) from images of street scenes, and generate new scenes that re-mix local patterns seamlessly. Fig. 9 displays the 3 training images and 8 generated images.

7. GENERATIVE CONVNET UNITS

In this section, we explain that a learned FRAME model based on ConvNet filters becomes a new ConvNet filter at the layer above the layer of filters employed by the model.

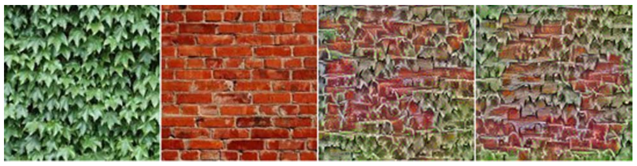


Figure 8. Generating hybrid texture patterns. The first 2 images are training images, and the last 2 images are generated images.

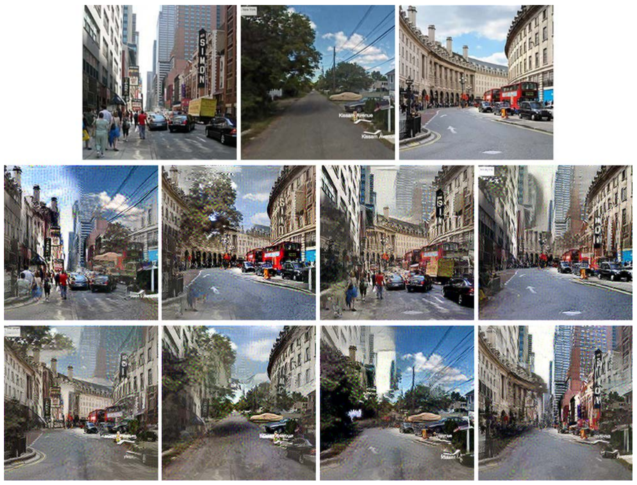


Figure 9. Generating scene patterns. The 3 images on the top row are training images, and the images on the bottom 2 rows are generated images.

We also explain the origin of the rectified linear transformation as an approximation to a mixture model of the presence and absence of the pattern modeled by the FRAME model. We then consider the learning of the generative model that involves a new layer of multiple filters to account for multiple local patterns in the non-aligned training images.

7.1 FRAME models as ConvNet units

On top of the convolutional layer of filters $\{F_k, k = 1, ..., K\}$, we can build another layer of filters $\{\mathbf{F}_j, j = 1, ..., J\}$ (with $\mathbf{F}$ in bold font, and indexed by j), so that

$$[\mathbf{F}_j * \mathbf{I}](y) = h\left(\sum_{k,x} w_{k,x}^{(j)} [F_k * \mathbf{I}](y+x) + b_j\right), \tag{17}$$

where $h()$ is a rectification function such as the rectified linear unit $h(r) = \max(0, r)$. Equation (17) follows the recursive equation (2) in Section 2. For notational simplicity, we make the indices of layers, l and $l-1$, implicit, and use bold font $\mathbf{F}$ and non-bold F to denote the filters at the two adjacent layers respectively. Also for simplicity, we ignore the layers of local max pooling and sub-sampling.

Model (9) corresponds to a single filter in $\{\mathbf{F}_j\}$ at a particular position y (e.g., the origin $y = 0$) where we assume that the object appears. The weights ($w^{(j)}_{k,x}$) can be learned by fitting model (9) using Algorithm 1, which enables us to add a ConvNet node in a generative fashion.

The log-likelihood ratio of the object model $p(\mathbf{I}; w)$ in (9) versus the background model $q(\mathbf{I})$ is

$$\log \frac{p(\mathbf{I}; w)}{q(\mathbf{I})} = \sum_k \sum_x w_{k,x}[F_k * \mathbf{I}](x) - \log Z(w). \tag{18}$$

It can be used as a score for detecting the object versus the background. If the score is below a threshold, no object is detected, and the score is rectified to 0. The rectified linear unit $h()$ in $\mathbf{F}_j$ in (17) accounts for the fact that at any position y, the object either appears or not. More formally, consider a mixture model $p(\mathbf{I}) = \alpha p(\mathbf{I}; w) + (1 - \alpha)q(\mathbf{I})$, where α is the frequency that the object is activated, and $1 - \alpha$ is the frequency of background. Then

$$\log \frac{p(\mathbf{I})}{q(\mathbf{I})} = \log \left[1 + \exp\left(\sum_k \sum_x w_{k,x}[F_k * \mathbf{I}](x) - \log Z(w) + \log \frac{\alpha}{1-\alpha}\right)\right] + \log(1-\alpha). \tag{19}$$

Now let us compare equation (19) to equation (17). We can approximate the soft max function $\log(1 + e^r)$ by the hard max function $h(r) = \max(0, r)$. Then we can identify the bias term as $b = \log(\alpha/(1 - \alpha)) - \log Z(w)$. Hence the rectified linear unit models a mixture of "on" and "off" of an object pattern.

7.2 Generative model with a new layer of filters

Model (9) is used to model images where the objects are aligned and are from the same category. For non-aligned images that may consist of multiple local patterns, we can extend model (9) to a convolutional version with multiple filters

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[\sum_{j=1}^{J} \sum_{x \in \mathcal{D}} [\mathbf{F}_j * \mathbf{I}](x)\right] q(\mathbf{I}), \tag{20}$$

where $\{\mathbf{F}_j\}$ are defined by (17). This model is a product of experts model [14], where each $[\mathbf{F}_j * \mathbf{I}](x)$ is an expert about a mixture of an activation or inactivation of a local pattern of type j at position x. We call model (20) with (17) the generative ConvNet model. The model can also be considered a dense version of the And-Or model [43], where the binary switch of each expert corresponds to an Or-node, and the product corresponds to an And-node.

The stationary model (10) corresponds to a special case of generative ConvNet model (20) with (17), where there is only one j, and $[\mathbf{F} * \mathbf{I}](x) = \sum_{k=1}^{K} w_k[F_k * \mathbf{I}](x)$, which is a special case of (17) without rectification. It is a 1×1 convolutional filter that combines lower layer filter responses at the same position.

More importantly, due to the recursive nature of ConvNet, if the weight parameters w_k of the stationary model (10) are absorbed into the filters F_k by multiplying the weight and bias parameters of each F_k by w_k, then the stationary model becomes the generative ConvNet model (20) except that the top-layer filters $\{\mathbf{F}_j\}$ are replaced by the lower layer filters $\{F_k\}$. The learning of the stationary model (10) is a simplified version of the learning of the generative ConvNet model (20) where there is only one multiplicative parameter w_k for each filter F_k. The learning of the stationary model (10) is more unsupervised and more indicative of the expressiveness of the ConvNet features than the learning of the non-stationary model (9) because the former does not require alignment.

7.3 EM-like learning with latent switch variables

Suppose we observe $\{\mathbf{I}_i, i = 1, ..., n\}$ from the generative ConvNet model (20) with (17). 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 (20) and (17), then

$$\frac{\partial L(w)}{\partial w^{(j)}_{k,x}} = \frac{1}{n}\sum_{i=1}^{n} \sum_{y \in \mathcal{D}} \delta_{j,y}(\mathbf{I}_i)[F_k * \mathbf{I}_i](y + x) - \mathrm{E}_w\left[\sum_{y \in \mathcal{D}} \delta_{j,y}(\mathbf{I})[F_k * \mathbf{I}](y + x)\right], \tag{21}$$

where

$$\delta_{j,y}(\mathbf{I}) = h'\left(\sum_{k,x} w^{(j)}_{k,x}[F_k * \mathbf{I}](y + x) + b_j\right) \tag{22}$$

is a binary on/off detector of the local pattern of type j at position y on image $\mathbf{I}$, because for $h(r) = \max(0, r)$, $h'(r) = 0$ if $r \leq 0$, and $h'(r) = 1$ if $r > 0$. The gradient (21) admits an EM [5] interpretation which is typical in unsupervised learning algorithms that involve latent variables. Specifically, $\delta_{j,y}()$ detects the local pattern of type j modeled by $\mathbf{F}_j$. This step can be considered a hard-decision E-step. With the local patterns detected, the parameters of $\mathbf{F}_j$ are then updated in a similar way as in (13), which can be considered the M-step. That is, we learn $\mathbf{F}_j$ only from image patches where we detect pattern j. Such a scheme was used by [16] to learn codebooks of active basis models [34].

Model (20) with (17) defines a recursive scheme, where the learning of higher layer filters $\{\mathbf{F}_j\}$ is based on the lower layer filters $\{F_k\}$. We can use this recursive scheme to build up the layers from scratch. We can start from the ground layer of the raw image, and learn the first layer filters. Then based on the first layer filters, we learn the second layer filters, and so on.

After building up the model layer by layer, we can continue to refine the parameters of all the layers simultaneously. We can also learn all the layers simultaneously from scratch. In fact, the parameter w in model (20) can be interpreted more broadly as multi-layer connection weights that define all the layers of filters. The gradient of the log-likelihood is

$$\frac{\partial L(w)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{J}\sum_{x\in\mathcal{D}}\frac{\partial}{\partial w}[\mathbf{F}_j * \mathbf{I}_i](x) - \mathrm{E}_w\left[\sum_{j=1}^{J}\sum_{x\in\mathcal{D}}\frac{\partial}{\partial w}[\mathbf{F}_j * \mathbf{I}](x)\right], \tag{23}$$

where $\partial[\mathbf{F}_j * \mathbf{I}](x)/\partial w$ involves multiple layers of binary detectors. The resulting algorithm also requires partial derivative $\partial[\mathbf{F}_j * \mathbf{I}](x)/\partial \mathbf{I}$ for Langevin sampling, which can be considered a recurrent generative model driven by the binary switches at multiple layers. Both $\partial[\mathbf{F}_j * \mathbf{I}](x)/\partial w$ and $\partial[\mathbf{F}_j * \mathbf{I}](x)/\partial \mathbf{I}$ are readily available via back-propagation. See [13, 27] for earlier work along this direction. See also [3] for generative gradient of ConvNet.

Finally, we can also learn a FRAME model based on the features at the top layer,

$$p(\mathbf{I}; w) = \frac{1}{Z(w)}\exp\left[\sum_{k=1}^{K} w_k[F_k * \mathbf{I}]\right] q(\mathbf{I}), \tag{24}$$

where F_k is the k-th feature at the top layer, K is the total number of features at this layer (e.g., $K = 4{,}096$), and w_i are the parameters, $w = (w_k, \forall k)$. F_k can still be viewed as a filter whose filter map is 1×1. Suppose there are a number of image categories, and suppose we learn a model (24) for each image category with a category-specific w. Also suppose we are given the prior frequency of each category. A simple exercise of the Bayes rule then gives us the soft-max classification rule for the posterior probability of category given image, which is the discriminative ConvNet defined by equation (4) in Section 2.

8. MORE IMAGE GENERATION EXPERIMENT

Experiment 4: learning from non-aligned images. We learn the generative ConvNet model (20) with (17). Fig. 10 displays 3 experiments. In each row, the first image is the training image, and the next 3 images are generated by the learned model. In the first scenery experiment, we learn 10 filters at the 4th convolutional layer (without local max pooling), based on the pre-trained VGG filters at the 3rd convolutional layer. The size of each Conv4 filter to be learned is $11 \times 11 \times 256$. In the sunflower and egret experiments, we learn 20 filters of size $7 \times 7 \times 256$ (with local max pooling). Clearly these learned filters capture the

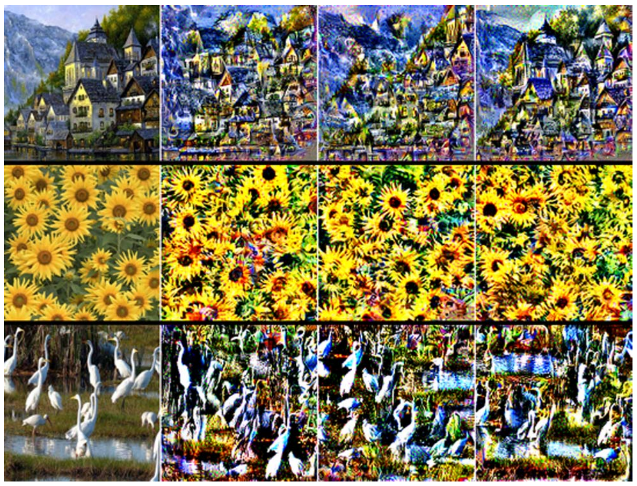


Figure 10. Learning without alignment. In each row, the first image is the training image, and the next 3 images are generated images.

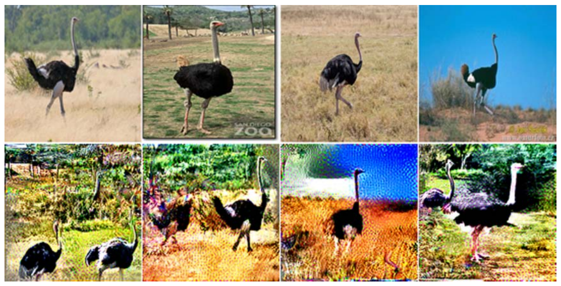


Figure 11. Learning from non-aligned image patterns. The first row displays 4 of the training images, and the second row displays generated images.

local patterns and re-shuffle them seamlessly. Fig. 11 displays an experiment where we learn the model from a small training set of non-aligned images. The first row displays 4 examples of training images and the second row displays the generated images. We use the same parameter setting as in the sunflower experiment. These experiments show that it is possible to learn generative ConvNet model (20) from non-aligned images.

9. LEARNING ALL LAYERS OF FILTERS

The method of learning a new layer of filters can be generalized to learning all layers of filters from scratch, without relying on a pre-trained ConvNet. Specifically, define the scoring function

$$f(\mathbf{I}; w) = \sum_{k=1}^{K}\sum_{x\in\mathcal{D}} w_{k,x}([\mathbf{F}_k * \mathbf{I}](x)), \tag{25}$$

and define the probability distribution of the model

$$p(\mathbf{I};w) = \frac{1}{Z(w)} \exp\left[f(\mathbf{I};w)\right] q(\mathbf{I}), \tag{26}$$

where this time w collects the weight and bias terms that define the filters at all the layers, instead of the weight and bias terms only at the top layer. Then w can be updated by gradient ascent. The gradient is

$$\frac{\partial L(w)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial w} f(\mathbf{I}_i;w) - \mathrm{E}_w\left[\frac{\partial}{\partial w} f(\mathbf{I};w)\right], \tag{27}$$

where again the expectation can be approximated by the Monte Carlo samples obtained by the Langevin dynamics which samples from $p(\mathbf{I};w)$. Updating w according to the above gradient will shift the density $p(\mathbf{I};w)$ from the synthesized images towards the observed images.

We learn a 5-layer non-stationary generative ConvNet model from 1,000 face images randomly sampled from the CelebA dataset [24]. Images are cropped at the center to 64×64. The first layer has $7 \times 7 \times 96$ filters (i.e., 96 filters, and the size of each filter is 7×7) with stride of 3 pixels, the second layer has $5 \times 5 \times 128$ filters with stride of 2 pixels, the third layer has $5 \times 5 \times 256$ filters with stride of 2 pixels, the fourth layer has $5 \times 5 \times 512$ filters with stride of 2 pixels, and the final layer is a fully connected layer that outputs the value of $f(\mathbf{I};w)$. In our learning algorithm, the number of learning iterations is 300, and the initial learning rate is 0.3. Learning rate is decayed logarithmically every 10 iterations. We continue to use the Langevin dynamics to generate the synthesized images. Specifically, within each iteration, for each training image, we obtain a corresponding synthesized image by running the Langevin dynamics from a constant image of the average intensity of the training image, i.e., we adopt the "cold start" scheme for Langevin sampling. We run 30 Langevin steps within each learning iteration, and the step size of the Langevin dynamics is 0.3. Fig. 12 shows random examples of the training images and the synthesized images generated by the learned model.

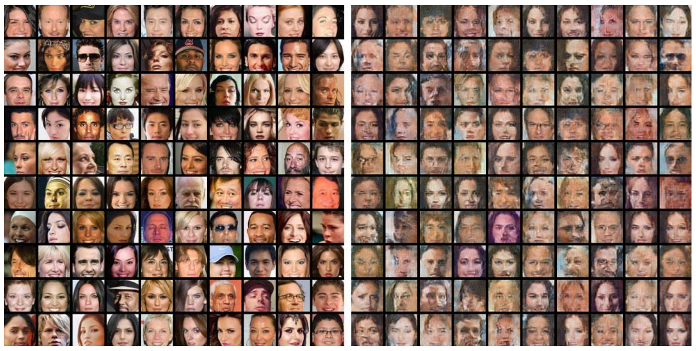


Figure 12. Images synthesized by the model learned on the CelebA dataset. The left panel displays random examples of the training images and the right panel displays random examples of the synthesized images generated by the learned model.

To monitor model fitting, we compute the means and standard deviations of the scoring function $f(\mathbf{I};w)$ (which is $\log[p(\mathbf{I};w)/q(\mathbf{I})] + \log Z(w)$, i.e., the log-likelihood ratio up to a constant) on the 1,000 face images and 1,000 randomly sampled natural images from 10 scene categories of ImageNet dataset [6]. Table 1 displays the results. Clearly the learned model puts higher probabilities on the face images than the random natural images. For the purpose of classification, we need to learn a model of natural images, in addition to the model of face images, and perform likelihood ratio test for classification. We shall investigate this issue in our future work.

Table 1. Means and standard deviations ($\times 10^6$) of the scoring function $f(\mathbf{I};w)$ on the 1,000 face images and 1,000 natural images

	face images	natural images
mean	6.35	4.59
std	0.75	1.20

We then test the learned model quantitatively on the image recovery task. On each testing image $\mathbf{I}$, we randomly place a squared mask M, which consists of all the masked pixels. We then recover the masked pixels by sampling from the conditional distribution $p(\mathbf{I}_M \mid \mathbf{I}_{\bar{M}}, w)$ according to the learned model $p(\mathbf{I};w)$, where $\bar{M}$ denotes the unmasked pixels, and $\mathbf{I}_M$ and $\mathbf{I}_{\bar{M}}$ are the masked part and the unmasked part of the image $\mathbf{I}$ respectively. The sampling of $p(\mathbf{I}_M \mid \mathbf{I}_{\bar{M}}, w)$ is again accomplished by the Langevin dynamics, which is the same as the Langevin dynamics that samples from the full distribution $p(\mathbf{I};w)$, except that we fix the unmasked part $\mathbf{I}_{\bar{M}}$ and only update the masked part $\mathbf{I}_M$ throughout the Langevin dynamics. In the learning stage, we learn the model from the fully observed training images. To specialize the learned model to this recovery task, we learn $p(\mathbf{I}_M \mid \mathbf{I}_{\bar{M}}, w)$ directly. That is, in the learning stage, we also randomly place a mark M on each fully observed training image $\mathbf{I}$, and run Langevin dynamics by fixing $\mathbf{I}_{\bar{M}}$ to obtain the synthesized image. The parameters w are then updated by gradient ascent according to (27).

After learning the model from 1,000 face images randomly sampled from the CelebA dataset, we randomly sample another 1,000 images for testing, and randomly place a mask M on each testing image $\mathbf{I}$. We then recover the masked pixels by sampling from $p(\mathbf{I}_M \mid \mathbf{I}_{\bar{M}}, w)$ using Langevin dynamics. In the training stage, we randomly place a 40×40 mask on each training image. In the testing stage, we experiment with two sizes of masks: 40×40 (M40) and 50×50 (M50). We compare our method with 8 different image recovery methods. MRF-ℓ_2 and MRF-ℓ_1 are based on Markov random field prior models where the nearest neighbor potential terms are ℓ_2 and ℓ_1 differences respectively. The rest (Nan1 - 6) are interpolation based methods. Please refer

Table 2. Image recovery results of different methods. The errors are measured by the average per-pixel differences between the original images and the recovered images on the masked pixels. The smaller the better

Methods	MRF-ℓ_2	MRF-ℓ_1	Nan1	Nan2	Nan3	Nan4	Nan5	Nan6	Ours
M40	0.1519	0.1507	0.1397	0.1363	0.3170	0.1366	0.1368	0.1397	**0.1338**
M50	0.1847	0.1841	0.1765	0.1690	0.3471	0.1671	0.1672	0.1765	**0.1465**

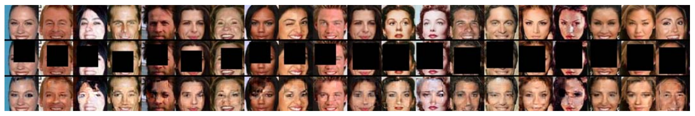


Figure 13. Image recovery. The first row displays the original testing images, the second row shows the corresponding masked testing images, and the third row displays the recovered images based on the learned model.

to [9] for more details. We measure the recovery error by the average of per-pixel differences between the original testing images and the corresponding recovered images on the masked pixels. Table 2 displays the numerical comparison results. Fig. 13 displays some examples of image recovery.

10. CONCLUSION

In this paper, we learn the FRAME models based on pre-trained ConvNet filters or features. Just as weighted summations of three basic colors of Red, Green, and Blue can generate any visible colors, the FRAME models that are based on weighted summations of these features can generate a wide variety of natural image patterns. The learned FRAME models themselves become new ConvNet units. We also show that it is possible to learn the multilayer FRAME model or the generative ConvNet model (20) from scratch without relying on pre-trained ConvNet filters.

CODE AND DATA

The code, data, and more experimental results can be found at http://www.stat.ucla.edu/~yang.lu/project/deepFrame/main.html.

APPENDIX A. MAXIMUM ENTROPY JUSTIFICATION

The FRAME model (9) can be justified by the maximum entropy or minimum divergence principle. Suppose the true distribution that generates the observed images $\{\mathbf{I}_i\}$ is $f(\mathbf{I})$. Let $w^\star$ solve the population version of the maximum likelihood equation:

$$\mathrm{E}_w([F_k * \mathbf{I}](x)) = \mathrm{E}_f([F_k * \mathbf{I}](x)),\ \forall k, x. \tag{28}$$

Let Ω be the set of all the feasible distributions p that share the statistical properties of f as captured by $\{F_k\}$:

$$\Omega = \{p : \mathrm{E}_p([F_k * \mathbf{I}](x)) = \mathrm{E}_f([F_k * \mathbf{I}](x))\ \forall k, x\}. \tag{29}$$

Then it can be shown that among all $p \in \Omega$, $p(\mathbf{I}; w^\star)$ achieves the minimum of $\mathrm{KL}(p||q)$, i.e., the Kullback-Leibler divergence from p to q [4]. Thus $p(\mathbf{I}; w^\star)$ can be considered the projection of q onto Ω, or the minimal modification of the reference distribution q to match the statistical properties of the true distribution f. In the special case where q is a uniform distribution, $p(\mathbf{I}; w^\star)$ achieves the maximum entropy among all distributions in Ω. For Gaussian white noise q, as mentioned before, we can absorb the $\frac{\|\mathbf{I}\|^2}{2\sigma^2}$ term into the energy function as in (15), so model (9) can be written relative to a uniform measure with $\|\mathbf{I}\|^2$ as an additional feature. The maximum entropy interpretation thus still holds if we opt to estimate σ^2 from the data.

APPENDIX B. JULESZ ENSEMBLE JUSTIFICATION

The learning algorithm seeks to match statistics of the synthesized images to those of the observed images, as indicated by (13) and (14), where the difference between the observed statistics and the synthesized statistics drives the update of the parameters. If the algorithm converges, and if the number of the synthesized images $\tilde{M}$ is large in the case of object patterns or if the image domain $\mathcal{D}$ is large in the case of texture patterns, then the synthesized statistics should match the observed statistics. Assume $q(\mathbf{I})$ to be the uniform distribution for now. We can consider the following ensemble in the case of object patterns:

$$\mathcal{J} = \Bigg\{(\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}) : \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}[F_k * \tilde{\mathbf{I}}_i](x) = \frac{1}{n}\sum_{i=1}^{n}[F_k * \mathbf{I}_i](x), \forall k, x\Bigg\}. \tag{30}$$

Consider the uniform distribution over $\mathcal{J}$. Then as $\tilde{n} \to \infty$, the marginal distribution of any $\tilde{\mathbf{I}}_i$ is given by model (9) with w being estimated by maximum likelihood. Conversely, model (9) puts uniform distribution on $\mathcal{J}$ if $\tilde{\mathbf{I}}_i$ are independent samples from model (9) and if $\tilde{n} \to \infty$.

As for the texture model, we can take $\tilde{n} = 1$, but let the image size go to ∞. First fix the square domain $\mathcal{D}$. Then embed it at the center of a larger square domain $\overline{\mathcal{D}}$. Consider the ensemble of images defined on $\overline{\mathcal{D}}$:

$$\mathcal{J} = \left\{\tilde{\mathbf{I}} : \frac{1}{|\overline{\mathcal{D}}|} \sum_{x \in \overline{\mathcal{D}}} [F_k * \tilde{\mathbf{I}}](x) = \frac{1}{|\mathcal{D}|} \frac{1}{n} \sum_{i=1}^{n} \sum_{x \in \mathcal{D}} [F_k * \mathbf{I}_i](x), \forall k \right\}. \tag{31}$$

Then under the uniform distribution on $\mathcal{J}$, as $|\overline{\mathcal{D}}| \to \infty$, the distribution of $\tilde{\mathbf{I}}$ restricted to $\mathcal{D}$ is given by model (10). Conversely, model (10) defined on $\overline{\mathcal{D}}$ puts uniform distribution on $\mathcal{J}$ as $|\overline{\mathcal{D}}| \to \infty$.

The ensemble $\mathcal{J}$ is called the Julesz ensemble by [36], because Julesz was the first to pose the question as to what statistics define a texture pattern [17]. The averaging across images in equation (30) enables re-mixing of the parts of the observed images to generate new object images. The spatial averaging in equation (31) enables re-shuffling of the local patterns in the observed image to generate a new texture image. That is, the averaging operations lead to exchangeability.

For object patterns, define the discrepancy

$$\Delta_{k,x} = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} [F_k * \tilde{\mathbf{I}}_i](x) - \frac{1}{n} \sum_{i=1}^{n} [F_k * \mathbf{I}_i](x). \tag{32}$$

One can sample from the uniform distribution on $\mathcal{J}$ in (30) by running a simulated annealing algorithm that samples from $p(\tilde{\mathbf{I}}_i, i = 1, ..., \tilde{n}) \propto \exp(-\sum_{k,x} \Delta_{k,x}^2 / T)$ by Langevin dynamics while gradually lowering the temperature T, or simply by gradient descent as in [10] by assuming $T = 0$. The sampling algorithm is very similar to Algorithm 1. One can use a similar method to sample from the uniform distribution over $\mathcal{J}$ in (31). Such a scheme was used by [42] for texture synthesis.

In the above discussion, we assume $q(\mathbf{I})$ to be the uniform distribution. If $q(\mathbf{I})$ is Gaussian, we only need to add the feature $\|\mathbf{I}\|^2$ to the pool of features to be matched. The above results still hold.

The Julesz ensemble perspective connects statistics matching and the FRAME models, thus providing another justification for these models in addition to the maximum entropy principle.

ACKNOWLEDGEMENTS

The code in our work is based on the Matlab code of MatConvNet [33], and our experiments are based on the VGG features [31].

We thank Jifeng Dai for earlier collaboration on generative ConvNet. We thank Junhua Mao and Zhuowen Tu for sharing their expertise on ConvNets.

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.

Received 12 May 2017

REFERENCES

[1] Bengio, Y., Goodfellow, I. J. and Courville, A. (2015). Deep Learning. MIT Press. MR3617773

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

[3] Dai, J., Lu, Y. and Wu, Y. N. (2015). Generative Modeling of Convolutional Neural Networks. In *Proceedings of International Conference on Learning Representations*.

[4] Della Pietra, S., Della Pietra, V. and Lafferty, J. (1997). Inducing features of random fields. *Pattern Analysis and Machine Intelligence* **19** 380–393.

[5] Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (Methodological)* 1–38. MR0501537

[6] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of Computer Vision and Pattern Recognition*.

[7] Denton, E., Chintala, S., Szlam, A. and Fergus, R. (2015). Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Proceedings of Advances in Neural Information Processing Systems*.

[8] Dosovitskiy, E., Springenberg, J. T. and Brox, T. (2015). Learning to Generate Chairs with Convolutional Neural Networks. In *Proceedings of Computer Vision and Pattern Recognition*.

[9] D'Errico and John (2012). Interpolation inpainting. https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint-nans.

[10] Gatys, L. A., Ecker, A. S. and Bethge, M. (2015). A Neural Algorithm of Artistic Style. *CoRR* **abs/1508.06576**.

[11] Geman, S. and Graffigne, C. (1986). Markov random field image models and their applications to computer vision. In *Proceedings of International Congress of Mathematicians* **1** 2.

[12] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J. and Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation. In *Proceedings of International Conference on Machine Learning* 1462–1471.

[13] Hinton, G., Osindero, S., Welling, M. and Teh, Y.-W. (2006). Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science* **30** 725–731.

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

[15] Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R. M. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science* **268** 1158–1161.

[16] Hong, Y., Si, Z. Z., Hu, W. Z., Zhu, S. C. and Wu, Y. N. (2014). Unsupervised Learning of Compositional Sparse Code for Natural Image Representation. *Quarterly of Applied Mathematics* **79** 373–406. MR3186243

[17] Julesz, B. (1962). Visual pattern discrimination. *IRE Transactions on Information Theory* **8** 84–92.

[18] Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proceedings of International Conference on Learning Representations*.

[19] Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems* 1097–1105.

[20] Kulkarni, T. D., Whitney, W. F., Kohli, P. and Tenenbaum, J. B. (2015). Deep Convolutional Inverse Graphics Network. In *Proceedings of International Conference on Neural Information Processing Systems* 2539–2547.

[21] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE* **86** 2278–2324.

[22] Liang, F., Liu, C. and Carroll, R. (2011). *Advanced Markov chain Monte Carlo methods: learning from past samples* **714**. John Wiley & Sons.

[23] Liu, J. S. (2008). *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

[24] Liu, Z., Luo, P., Wang, X. and Tang, X. (2015). Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision*.

[25] Lu, Y., Zhu, S.-C. and Wu, Y. N. (2016). Learning FRAME Models Using CNN Filters. In *AAAI*.

[26] Mnih, A. and Gregor, K. (2014). Neural Variational Inference and Learning in Belief Networks. In *Proceedings of International Conference on Machine Learning* 1791–1799.

[27] Ngiam, J., Chen, Z., Koh, P. W. and Ng, A. Y. (2011). Learning Deep Energy Models. In *Proceedings of International Conference on Machine Learning* 1105–1112.

[28] Rezende, D. J., Mohamed, S. and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of International Conference on Machine Learning* 1278–1286. JMLR Workshop and Conference Proceedings.

[29] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* **115** 211–252.

[30] Simonyan, K., Vedaldi, A. and Zisserman, A. (2015). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proceedings of International Conference on Learning Representations*.

[31] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of International Conference on Learning Representations*.

[32] Srivastava, A., Lee, A. B., Simoncelli, E. P. and Zhu, S.-C. (2003). On advances in statistical modeling of natural images. *Journal of Mathematical Imaging and Vision* **18** 17–33.

[33] Vedaldi, A. and Lenc, K. (2014). MatConvNet – Convolutional Neural Networks for MATLAB. *CoRR* **abs/1412.4564**.

[34] Wu, Y. N., Si, Z., Gong, H. and Zhu, S.-C. (2010). Learning active basis model for object detection and recognitio. *International Journal of Computer Vision* **90** 198–235.

[35] Wu, Y. N., Xie, J., Lu, Y. and Zhu, S.-C. Sparse and deep generalizations of the FRAME model. *Annals of Mathematical Sciences and Applications*.

[36] Wu, Y. N., Zhu, S.-C. and Liu, X. (2000). Equivalence of Julesz ensembles and FRAME models. *International Journal of Computer Vision* **38** 247–265.

[37] Xie, J., Hu, W., Zhu, S.-C. and Wu, Y. N. (2015). Learning sparse FRAME models for natural image patterns. *International Journal of Computer Vision* **114** 91–112.

[38] Xie, J., Lu, Y., Zhu, S.-C. and Wu, Y. N. (2015). Inducing wavelets into random fields via generative boosting. *Journal of Applied and Computational Harmonic Analysis* **41** 4–25.

[39] Younes, L. (1999). On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes* **65** 177–228.

[40] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional neural networks. In *Proceedings of European Conference on Computer Vision*.

[41] Zhu, S. C. and Mumford, D. B. (1998). GRADE: Gibbs Reaction And Diffusion Equations. In *Proceedings of International Conference on Computer Vision*.

[42] Zhu, S. C., Liu, X. and Wu, Y. N. (2000). Exploring texture ensembles by efficient Markov chain Monte Carlo - towards a 'trichromacy' theory of texture. *Pattern Analysis and Machine Intelligence* **22** 245–261.

[43] Zhu, S. C. and Mumford, D. (2006). A Stochastic Grammar of Images. *Foundations and Trends in Computer Graphics and Vision* **2** 259–362.

[44] Zhu, S. C., Wu, Y. N. and Mumford, D. (1997). Minimax entropy principle and its application to texture modeling. *Neural Computation* **9** 1627–1660.

Yang Lu
UCLA Department of Statistics
Los Angeles, CA
90095-1554
USA
E-mail address: yanglv@ucla.edu

Ruiqi Gao
UCLA Department of Statistics
Los Angeles, CA
90095-1554
USA
E-mail address: ruiqigao@ucla.edu

Song-Chun Zhu
UCLA Department of Statistics
Los Angeles, CA
90095-1554
USA
E-mail address: sczhu@stat.ucla.edu

Ying Nian Wu
UCLA Department of Statistics
Los Angeles, CA
90095-1554
USA
E-mail address: ywu@stat.ucla.edu