Learning FRAME Models Using CNN Filters

Yang Lu, Song-Chun Zhu, Ying Nian Wu
Department of Statistics, University of California, Los Angeles, USA

Abstract

The convolutional neural network (ConvNet or CNN) has proven to be very successful in many tasks such as those in computer vision. In this conceptual paper, we study the generative perspective of the discriminative CNN. In particular, we propose to learn the generative FRAME (Filters, Random field, And Maximum Entropy) model using the highly expressive filters pre-learned by the CNN at the convolutional layers. We show that the learning algorithm can generate realistic and rich object and texture patterns in natural scenes. We explain that each learned model corresponds to a new CNN unit at a layer above the layer of filters employed by the model. We further show that it is possible to learn a new layer of CNN units using a generative CNN model, which is a product of experts model, and the learning algorithm admits an EM interpretation with binary latent variables.

1 Introduction

The breakthrough made by the convolutional neural network (ConvNet or CNN) (Krizhevsky, Sutskever, and Hinton, 2012; LeCun et al., 1998) on the ImageNet dataset (Deng et al., 2009) was a watershed event that has transformed the fields of computer vision and speech recognition as well as related industries. While CNN has proven to be a powerful discriminative machine, researchers have recently become increasingly interested in the generative perspective of CNN. An interesting example is the recent work of Google deep dream (http://deepdreamgenerator.com/). Although it did not smash any performance records, it did capture people's imagination by generating interestingly vivid images.

In this conceptual paper, we explore the generative perspective of CNN more formally by defining generative models based on CNN features, and learning these models by generating images from the models. Adopting the metaphor of Google deep dream, we let the generative models dream by generating images. But unlike the google deep dream, we learn the models from real images by making the dreams come true.

Specifically, we propose to learn the FRAME (Filters, Random field, And Maximum Entropy) models (Zhu, Wu, and Mumford, 1997; Xie et al., 2015a) using the highly non-linear filters pre-learned by CNN at the convolutional layers. A FRAME model is a random field model that defines a probability distribution on the image space. 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 (Zhu, Wu, and Mumford, 1997). The more recent version is a non-stationary extension designed to represent object patterns (Xie et al., 2015a). 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 model are the oriented and elongated Gabor filters at different scales, as well as the isotropic Difference of Gaussian (DoG) filters of different sizes. These are linear filters that capture simple local image features such as edges and blobs. With the emergence of the more expressive nonlinear filters learned by CNN at various convolutional layers, it is only natural to replace the linear filters in the original FRAME models by the CNN filters in the hope of learning more expressive models.

We use the Langevin dynamics to sample from the probability distribution defined by the model. Such a dynamics was first applied to the FRAME model by Zhu and Mumford (1998), 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 CNN filters, the dynamics can be viewed as a recurrent generative form of the model, where the reactions and diffusions are governed by the CNN filters of positive and negative weights respectively.

Incorporating CNN filters into the FRAME model is not an ad hoc utilitarian exploit. It is actually a seamless meshing between the FRAME model and the CNN model. The original FRAME model has an energy function that consists of a layer of linear filtering followed by a layer of point-

wise nonlinear transformation. It is natural to follow the deep learning philosophy to add alternative layers of linear filtering and nonlinear transformation to have a deep FRAME model that directly corresponds to a CNN. More importantly, the learned FRAME model using CNN filters corresponds to a new CNN unit at the layer directly above the layer of CNN filters employed by the FRAME model. In particular, the non-stationary FRAME becomes a single CNN node at a specific position where the object appears, whereas the stationary FRAME becomes a special type of convolutional unit. Therefore, the learned FRAME model can be viewed as a generative version of CNN unit.

In addition to learning a single CNN unit, we can also learn 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 CNN model. It is a product of experts model (Hinton, 2002), where each expert models a mixture of activation and inactivation of a local pattern. The rectified linear unit can be justified as an approximation to the energy function of this mixture model. The learning algorithm admits an interpretation in terms of the EM algorithm (Dempster, Laird, and Rubin, 1977) with a hard-decision E-step that detects the local patterns modeled by the convolutional units.

The main purpose of this paper is to establish the conceptual correspondence between the generative FRAME model and the discriminative CNN, thus providing a formal generative perspective for CNN. Such a perspective is much needed because it may eventually lead to unsupervised learning of CNN in a generative fashion without the need for image labeling.

2 Past work

Recently there have been many interesting papers on visualizing CNN nodes, such as deconvolutional networks (Zeiler and Fergus, 2014), score maximization (Simonyan, Vedaldi, and Zisserman, 2015), and the recent artful work of Google deep dream (http://deepdreamgenerator.com/) and painting style (Gatys, Ecker, and Bethge, 2015). Our work is different from these previous methods in that we learn a rigorously defined generative model from training images, and the learned models correspond to new CNN units. This work is a continuation of the recent work on generative CNN (Dai, Lu, and Wu, 2015).

There have also been recent papers on generative models based on supervised image generation (Dosovitskiy, Springenberg, and Brox, 2015), variational auto-encoders (Hinton et al., 1995; Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014; Mnih and Gregor, 2014; Kulkarni et al., 2015; Gregor et al., 2015), and adversarial networks (Denton et al., 2015). 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 weight parameters.

3 FRAME models based on linear filters

This section reviews the background on the FRAME models based on linear filters.

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. Let $F_k * \mathbf{I}$ be the filtered image or feature map, and $[F_k * \mathbf{I}](x)$ be the filter response at position x (x is a two-dimensional coordinate). A linear filter F_k can be written as a two-dimensional function $F_k(x)$, so that $[F_k * \mathbf{I}](y) = F_k(x)\mathbf{I}(y+x)$, which is a translation invariant linear operation.

The original FRAME model (Zhu, Wu, and Mumford, 1997) for texture patterns is a stationary or spatially homogeneous Markov random field or Gibbs distribution of the following form:

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

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 Zhu, Wu, and Mumford (1997), 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 filter map $F_k * \mathbf{I}$. The spatially pooled marginal histograms are the sufficient statistics of model (1).

Model (1) is stationary because the function $\lambda_k()$ does not depend on position x. This stationary model is used to model texture patterns. In model (1), 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 pointwise nonlinear transformation by $\{\lambda_k()\}$. Repeating this pattern recursively (while also adding local max pooling and subsampling) will lead to a generative version of CNN.

The non-stationary or spatially inhomogeneous FRAME model for object patterns (Xie et al., 2015a) is of the following form:

$$p(\mathbf{I}; \lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{k=1}^{K} \sum_{x \in \mathcal{D}} \lambda_{k,x}([F_k * \mathbf{I}](x))\right] q(\mathbf{I}), \quad (2)$$

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

$$\lambda_{k,x}(r) = w_{k,x} h(r), \quad (3)$$

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 Xie et al. (2015a) , they use $h(r) = |r|$

for full wave rectification. One can also use rectified linear unit $h(r) = \max(0, r)$ (Krizhevsky, Sutskever, and Hinton, 2012) 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

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{|\mathcal{D}|/2}} \exp\left[-\frac{1}{2\sigma^2}||\mathbf{I}||^2\right], \quad (4)$$

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

In the original FRAME model (1), $q(\mathbf{I})$ is assumed to be a uniform measure. In model (2), 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 (1). 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 (2) and (3), $(w_{k,x}, \forall x, k)$ can be considered a second-layer linear filter on top of the first layer filters $\{F_k\}$ rectified by $h()$.

Both models (1) and (2) can be sparsified. Model (1) can be sparsified by selecting a small set of filters F_k using the filter pursuit procedure (Zhu, Wu, and Mumford, 1997). Model (2) 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 (Xie et al., 2015a) or a generative boosting method (Xie et al., 2015b).

4 FRAME models based on CNN filters

Instead of using linear filters, we can use the filters at various convolutional layers of a pre-learned CNN. 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 CNN. 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 $h(r) = \max(0, r)$. Then the non-stationary FRAME model becomes

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

where $q(\mathbf{I})$ is again the Gaussian white noise model (4) , 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 (5) shares the same form as model (2) with linear filters, except that the rectification function $h()$ in model (2) is already absorbed in the CNN filers $\{F_k\}$ in model (5) with $h(r) = \max(0, r)$. We shall use model (5) for generating object patterns.

The stationary FRAME model is of the following form:

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

which is almost the same as model (5) except that w_k is the same across x. $w = (w_k, \forall k)$. We shall use model (6) for generating texture patterns.

Again, both models (5) and (6) can be sparsified, either by forward selection such as filter pursuit (Zhu, Wu, and Mumford, 1997) or generative boosting (Xie et al., 2015b), or by backward elimination.

5 Learning and sampling algorithm

The basic learning algorithm for object model estimates the unknown parameters w from a set of aligned training images $\{\mathbf{I}_m, m = 1, ..., M\}$ that come from the same object category, where M is the total number of training images. In the basic learning algorithm, the weight parameters w can be estimated by maximizing the log-likelihood function

$$L(w) = \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{I}_m; w), \quad (7)$$

where $p(\mathbf{I}; w)$ is defined by (5). $L(w)$ is a concave function. The first derivatives of $L(w)$ are

$$\frac{\partial L(w)}{\partial w_{k,x}} = \frac{1}{M}\sum_{m=1}^{M}[F_k * \mathbf{I}_m](x) - \mathrm{E}_w\left([F_k * \mathbf{I}](x)\right), \quad (8)$$

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 (Younes, 1999):

$$w_{k,x}^{(t+1)} = w_{k,x}^{(t)} + \gamma\left[\frac{1}{M}\sum_{m=1}^{M}[F_k * \mathbf{I}_m](x) - \frac{1}{\tilde{M}}\sum_{m=1}^{\tilde{M}}[F_k * \tilde{\mathbf{I}}_m](x)\right] \quad (9)$$

for every $k \in \{1, ..., K\}$ and $x \in \mathcal{D}$, where γ is the learning rate, and $\{\tilde{\mathbf{I}}_m\}$ are the synthesized images sampled from $p(\mathbf{I}; w^{(t)})$ using MCMC. $\tilde{M}$ 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 $\{[F_k * \mathbf{I}_m](x), \forall m\}$, as well as being inversely proportional to the iteration t as in stochastic approximation.

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

$$U(\mathbf{I}, w) = -\sum_{k=1}^{K}\sum_{x\in\mathcal{D}} w_{k,x}[F_k * \mathbf{I}](x) + \frac{1}{2\sigma^2}||\mathbf{I}||^2. \quad (10)$$

The Langevin dynamics iterates

$$\mathbf{I}_{\tau+1} = \mathbf{I}_\tau - \frac{\epsilon^2}{2}U'(\mathbf{I}_\tau, w) + \epsilon Z_\tau, \quad (11)$$

where $U'(\mathbf{I}, w) = \partial U(\mathbf{I}, w)/\partial \mathbf{I}$. This gradient can be computed by back-propagation. In (11), 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 Zhu and Mumford (1998), who called the corresponding gradient descent algorithm the Gibbs reaction and diffusion equations (GRADE).

Algorithm 1 Learning and sampling algorithm

Input:

(1) training images $\{\mathbf{I}_m, m = 1, ..., M\}$
(2) a filter bank $\{F_k, k = 1, ..., K\}$
(3) number of synthesized images $\tilde{M}$
(4) number of Langevin steps L
(5) number of learning iterations T

Output:

(1) estimated parameters $w = (w_{k,x}, \forall k, x)$
(2) synthesized images $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$

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

Algorithm 1 describes the details of the learning and sampling algorithm. Algorithm 1 embodies the principle of "analysis by synthesis," i.e., we generate synthesized images from the current model, and then update the model parameters based on the difference between the synthesized images and the observed images. If we consider the synthesized images as "dreams" of the current model (following the metaphor used by Google deep dream), then the learning algorithm is to make the dreams come true.

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. 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.

For learning stationary FRAME (6), usually $M = 1$, i.e., we observe one texture image, and we update the parameters

$$w^{(t+1)}_k = w^{(t)}_k + \frac{\gamma}{|\mathcal{D}|}\left[\frac{1}{M}\sum_{m=1}^{M}\sum_{x\in\mathcal{D}}[F_k * \mathbf{I}_m](x) - \frac{1}{\tilde{M}}\sum_{m=1}^{\tilde{M}}\sum_{x\in\mathcal{D}}[F_k * \tilde{\mathbf{I}}_m](x)\right] \tag{12}$$

for every $k \in \{1, ..., K\}$, where there is a spatial pooling across positions $x \in \mathcal{D}$. The sampling is again accomplished by Langevin dynamics. The learning and sampling algorithm for the stationary model (6) only involves minor modifications of Algorithm 1.

6 Generative CNN 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

$$[\mathbf{F}_j * \mathbf{I}](y) = h\left(\sum_{k,x} w^{(j)}_{k,x}[F_k * \mathbf{I}](y + x) + b_j\right), \tag{13}$$

where $h()$ is a rectification function such as the rectified linear unit $h(r) = \max(0, r)$, and where the bias term b_j is related to $-\log Z(w)$. For simplicity, we ignore the layers of local max pooling and sub-sampling.

Model (5) 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 (5) using Algorithm 1, which enables us to add a CNN node in a generative fashion.

The log-likelihood ratio of the object model $p(\mathbf{I}; w)$ in (5) versus the background model $q(\mathbf{I})$ is $\log(p(\mathbf{I}; w)/q(\mathbf{I})) = \sum_k \sum_x w_{k,x}[F_k * \mathbf{I}](x) - \log Z(w)$. 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 (13) 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. $\log(p(\mathbf{I})/q(\mathbf{I})) = \log(1 + \exp(\sum_k \sum_x w_{k,x}(F_k * \mathbf{I})(x) - \log Z(w) + \log(\alpha/(1 - \alpha)) + \log(1 - \alpha)$. We can approximate the soft max function $\log(1 + e^r)$ by the hard max function $\max(0, r)$. Thus we can identify the bias term as $b = \log(\alpha/(1 - \alpha)) - \log Z(w)$, and the rectified linear unit models a mixture of "on" and "off" of an object pattern.

Model (5) 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 (5) to a convolutional version with multiple filters

$$p(\mathbf{I}; w) = \frac{1}{Z(w)}\exp\left[\sum_{j=1}^{J}\sum_{x\in\mathcal{D}}[\mathbf{F}_j * \mathbf{I}](x)\right]q(\mathbf{I}), \tag{14}$$

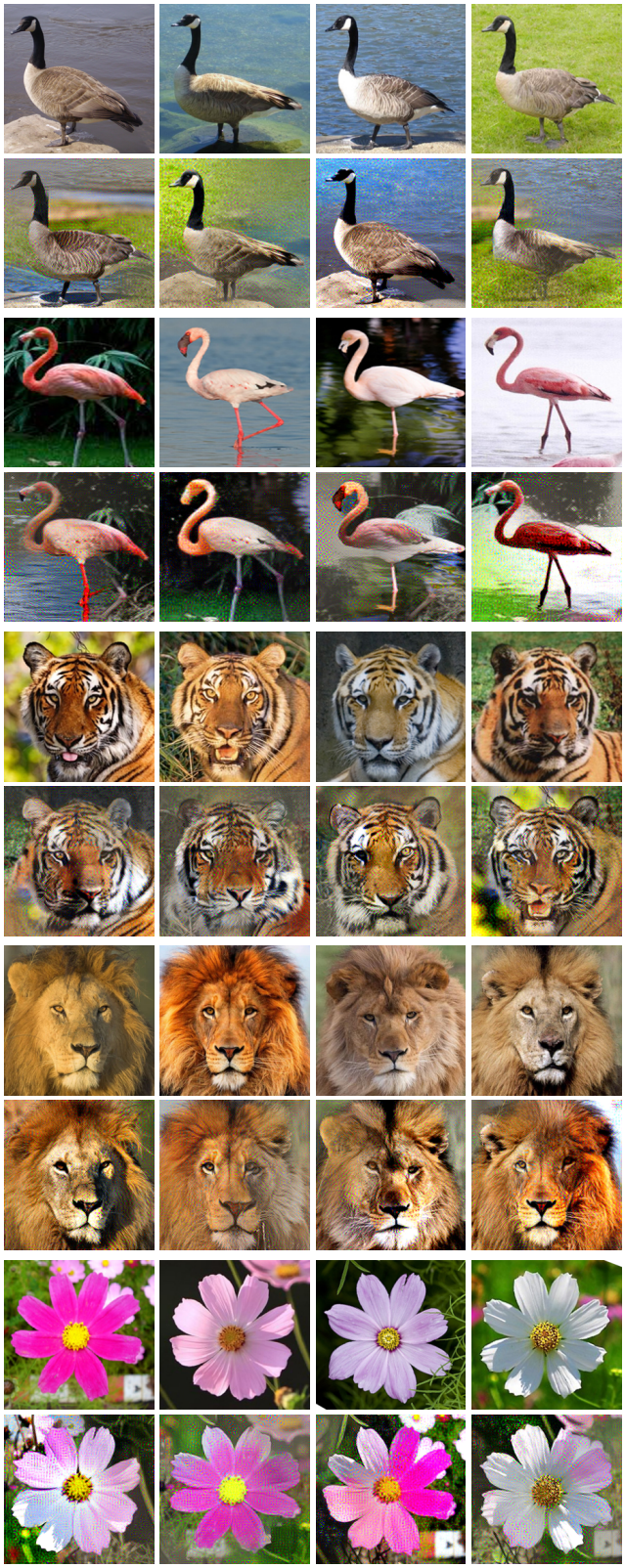

Figure 1: Generating object patterns. For each category, the first row displays 4 of the training images, and the second row displays generated images.

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

Figure 3: Generating hybrid object patterns. For each experiment, the first row displays 4 of the training images, and the second row displays generated images.

where $\{\mathbf{F}_j\}$ are defined by (13). This model is a product of experts model (Hinton, 2002), 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 (14) with (13) the generative CNN model. The model can also be considered a dense version of the And-Or model (Zhu and Mumford, 2006), where the binary switch of each expert corresponds to an Or-node, and the product corresponds to an And-node.

The stationary model (6) corresponds to a special case of generative CNN model (14) with (13), 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 (13) without rectification. It is a singleton filter that combines lower layer filter responses at the same position.

More importantly, due to the recursive nature of CNN, if the weight parameters w_k of the stationary model (6) 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 CNN model (14) except that the top-layer filters $\{\mathbf{F}_j\}$ are replaced by the lower layer filters $\{F_k\}$. The learning of the stationary model (6) is a simplified version of the learning of the generative CNN model (14) where there is only one multiplicative parameter w_k for each filter F_k. The learning of the stationary model (6) is more unsupervised and more indicative of the expressiveness of the CNN features than the learning of the non-stationary model (5) because the former does not require alignment.

Suppose we observe $\{\mathbf{I}_m, m = 1, ..., M\}$ from the generative CNN model (14) with (13). Let $L(w) = \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{I}_m; w)$ be the log-likelihood where $p(\mathbf{I}; w)$ is defined by (14) and (13), then

$$\frac{\partial L(w)}{\partial w_{k,x}^{(j)}} = \frac{1}{M}\sum_{m=1}^{M}\sum_{y\in\mathcal{D}} \delta_{j,y}(\mathbf{I}_m)[F_k * \mathbf{I}_m](y+x) - \mathrm{E}_w\left[\sum_{y\in\mathcal{D}} \delta_{j,y}(\mathbf{I})[F_k * \mathbf{I}](y+x)\right], \tag{15}$$

where

$$\delta_{j,y}(\mathbf{I}) = h'\left(\sum_{k,x} w_{k,x}^{(j)}[F_k * \mathbf{I}](y+x) + b_j\right) \tag{16}$$

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 (15) admits an EM (Dempster, Laird, and Rubin, 1977) 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 (9), 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 Hong et al. (2014) to learn codebooks of active basis models (Wu et al., 2010).

Model (14) with (13) 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. In fact, the parameter w in model (14) can be interpreted more broadly as multi-layer connection weights that define all the layers of filters. The gradient of the log-likelihood is

$$\frac{\partial L(w)}{\partial w} = \frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{J}\sum_{x\in\mathcal{D}} \frac{\partial}{\partial w}[\mathbf{F}_j * \mathbf{I}_m](x) - \mathrm{E}_w\left[\sum_{j=1}^{J}\sum_{x\in\mathcal{D}} \frac{\partial}{\partial w}[\mathbf{F}_j * \mathbf{I}](x)\right], \tag{17}$$

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

Figure 4: Generating hybrid texture patterns. The first 2 images are training images, and the last 2 images are generated images.

Figure 5: Learning without alignment. In each row, the first image is the training image, and the next 2 images are generated images.

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 Hinton et al. (2006); Ngiam et al. (2011) for earlier work along this direction. See also Dai, Lu, and Wu (2015) for generative gradient of CNN.

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

$$p(\mathbf{I}; W) = \frac{1}{Z(W)} \exp\left[\sum_{i=1}^{N} W_i[\Psi_i * \mathbf{I}]\right] q(\mathbf{I}), \qquad (18)$$

where Ψ_i is the i-th feature at the top fully connected layer, N is the total number of features at this layer (e.g., $N = 4096$), and W_i are the parameters, $W = (W_i, \forall i)$. Ψ_i 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 (18) 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 CNN.

7 Image generation experiments

In our experiments, we use VGG filters (Simonyan and Zisserman, 2015), and we use the Matlab code of MatConvNet (Vedaldi and Lenc, 2014).

Experiment 1: generating object patterns. We learn the non-stationary FRAME model (5) 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 $\tilde{M} = 16$ parallel chains for Langevin sampling. The number of Langevin iterations between every two consecutive updates of the parameters is $L = 100$. Fig. 1 shows some experiments using filters from the 3rd convolutional layer of VGG. 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.

Figure 6: Learning from non-aligned images. The first row displays 3 of the training images, and the second row displays 3 generated images.

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

Experiment 3: generating hybrid patterns. We learn models (5) and (6) from images of mixed categories, and generate hybrid patterns. Figs. 3 and 4 display a few examples.

Experiment 4: learning a new layer of filters from non-aligned images. We learn the generative CNN model (14) with (13). Fig. 5 displays 3 experiments. In each row, the first image is the training image, and the next 2 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 local patterns and re-shuffle them seamlessly. Fig. 6 displays an experiment where we learn a layer of filters from a small training set of non-aligned images. The first row displays 3 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 CNN model (14) from non-aligned images.

8 Conclusion

In this paper, we learn the FRAME models based on pre-trained CNN filters. It is possible to learn the multi-layer FRAME model or the generative CNN model (14) from scratch in a layer by layer fashion without relying on pre-trained CNN filters. The learning will be a recursion of model (14) , and it can be unsupervised without image labeling.

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`

Acknowlegements

The code in our work is based on the Matlab code of MatConvNet (Vedaldi and Lenc, 2014), and our experiments are based on the VGG features (Simonyan and Zisserman, 2015). We are grateful to these authors for sharing their code and results with the community.

We thank Jifeng Dai for earlier collaboration on generative CNN. We thank Junhua Mao and Zhuowen Tu for sharing their expertise on CNN. We thank one reviewer for deeply insightful comments. The work is supported by NSF DMS 1310391, ONR MURI N00014-10-1-0933, DARPA SIMPLEX N66001-15-C-4035, and DARPA MSEE FA 8650-11-1-7149.

Appendix 1. Maximum entropy justification

The FRAME model (5) can be justified by the maximum entropy or minimum divergence principle. Suppose the true distribution that generates the observed images $\{\mathbf{I}_m\}$ is $f(\mathbf{I})$. Let $w^\star$ solve the population version of the maximum likelihood equation:

$$\mathrm{E}_w([F_k * \mathbf{I}](x)) = \mathrm{E}_f([F_k * \mathbf{I}](x)),\ \forall k, x. \tag{19}$$

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

$$\Omega = \{p : \mathrm{E}_p([F_k * \mathbf{I}](x)) = \mathrm{E}_f([F_k * \mathbf{I}](x))\ \forall k, x\}. \tag{20}$$

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 (Della Pietra, Della Pietra, and Lafferty, 1997). 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 (10), so model (5) 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 2. Julesz ensemble justification

The learning algorithm seeks to match statistics of the synthesized images to those of the observed images, as indicated by (9) and (12), 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:

$$\mathcal{J} = \Big\{ (\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}) : \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} [F_k * \tilde{\mathbf{I}}_m](x) = \frac{1}{M} \sum_{m=1}^{M} [F_k * \mathbf{I}_m](x), \forall k, x \Big\}. \tag{21}$$

Consider the uniform distribution over $\mathcal{J}$. Then as $\tilde{M} \to \infty$, the marginal distribution of any $\tilde{\mathbf{I}}_m$ is given by model (5) with w being estimated by maximum likelihood. Conversely, model (5) puts uniform distribution on $\mathcal{J}$ if $\tilde{\mathbf{I}}_m$ are independent samples from model (5) and if $\tilde{M} \to \infty$.

As for the texture model, we can take $\tilde{M} = 1$, 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 $\overline{\mathcal{D}}$:

$$\mathcal{J} = \Big\{ \tilde{\mathbf{I}} : \frac{1}{|\overline{\mathcal{D}}|} \sum_{x \in \overline{\mathcal{D}}} [F_k * \tilde{\mathbf{I}}](x) = \frac{1}{|\mathcal{D}|} \frac{1}{M} \sum_{m=1}^{M} \sum_{x \in \mathcal{D}} [F_k * \mathbf{I}_m](x), \forall k \Big\}. \tag{22}$$

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 (6). Conversely, model (6) 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 Wu, Zhu, and Liu (2000), because Julesz was the first to pose the question as to what statistics define a texture pattern (Julesz, 1962). The averaging across images in equation (21) enables re-mixing of the parts of the observed images to generate new object images. The spatial averaging in equation (22) 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

$$\Delta_{k,x} = \frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} [F_k * \tilde{\mathbf{I}}_m](x) - \frac{1}{M} \sum_{m=1}^{M} [F_k * \mathbf{I}_m](x). \tag{23}$$

One can sample from the uniform distribution on $\mathcal{J}$ in (21) by running a simulated annealing algorithm that samples from $p(\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}) \propto \exp(-\sum_{k,x} \Delta_{k,x}^2 / T)$ by Langevin dynamics while gradually lowering the temperature T, or simply by gradient descent as in Gatys, Ecker,

and Bethge (2015) 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 (22). Such a scheme was used by Zhu, Liu, and Wu (2000) 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.

References

Dai, J.; Lu, Y.; and Wu, Y. N. 2015. Generative modeling of convolutional neural networks. In *ICLR*.

Della Pietra, S.; Della Pietra, V.; and Lafferty, J. 1997. Inducing features of random fields. *PAMI* 19(4):380–393.

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.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255. IEEE.

Denton, E.; Chintala, S.; Szlam, A.; and Fergus, R. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. *ArXiv e-prints*.

Dosovitskiy, E.; Springenberg, J. T.; and Brox, T. 2015. Learning to generate chairs with convolutional neural networks. In *CVPR*.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2015. A neural algorithm of artistic style. *ArXiv e-prints*.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D. J.; and Wierstra, D. 2015. DRAW: A recurrent neural network for image generation. In *ICML*, 1462–1471.

Hinton, G.; Dayan, P.; Frey, B. J.; and Neal, R. M. 1995. The wake-sleep algorithm for unsupervised neural networks.

Hinton, G.; Osindero, S.; Welling, M.; and Teh, Y.-W. 2006. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science* 30(4):725–731.

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

Hong, Y.; Si, Z.; Hu, W.; Zhu, S.; and Wu, Y. 2014. Unsupervised learning of compositional sparse code for natural image representation. *Quarterly of Applied Mathematics* 79:373–406.

Julesz, B. 1962. Visual pattern discrimination. *IRE Transactions on Information Theory* 8(2):84–92.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. *ICLR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Kulkarni, T. D.; Whitney, W.; Kohli, P.; and Tenenbaum, J. B. 2015. Deep Convolutional Inverse Graphics Network. *ArXiv e-prints*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. In *ICML*.

Ngiam, J.; Chen, Z.; Koh, P. W.; and Ng, A. Y. 2011. Learning deep energy models. In *ICML*.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. In Jebara, T., and Xing, E. P., eds., *ICML*, 1278–1286. JMLR Workshop and Conference Proceedings.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2015. Deep inside convolutional networks: Visualising image classification models and saliency maps. *ICLR*.

Vedaldi, A., and Lenc, K. 2014. Matconvnet – convolutional neural networks for matlab. *CoRR* abs/1412.4564.

Wu, Y. N.; Si, Z.; Gong, H.; and Zhu, S.-C. 2010. Learning active basis model for object detection and recognitio. *IJCV* 90:198–235.

Wu, Y. N.; Zhu, S.-C.; and Liu, X. 2000. Equivalence of julesz ensembles and frame models. *IJCV* 38:247–265.

Xie, J.; Hu, W.; Zhu, S.-C.; and Wu, Y. N. 2015a. Learning sparse frame models for natural image patterns. *IJCV* 114:91–112.

Xie, J.; Lu, Y.; Zhu, S.-C.; and Wu, Y. N. 2015b. Inducing wavelets into random fields via generative boosting. *Journal of Applied and Computational Harmonic Analysis*.

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(3-4):177–228.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional neural networks. *ECCV*.

Zhu, S., and Mumford, D. 1998. Grade: Gibbs reaction and diffusion equations. In *ICCV*.

Zhu, S. C., and Mumford, D. 2006. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision* 2(4):259–362.

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. *PAMI* 22:245–261.

Zhu, S. C.; Wu, Y. N.; and Mumford, D. 1997. Minimax entropy principle and its application to texture modeling. *Neural Computation* 9(8):1627–1660.