QUARTERLY OF APPLIED MATHEMATICS
VOLUME , NUMBER 0
XXXX XXXX, PAGES 000–000

A TALE OF THREE PROBABILISTIC FAMILIES: DISCRIMINATIVE, DESCRIPTIVE AND GENERATIVE MODELS

BY

YING NIAN WU (*Department of Statistics, University of California, Los Angeles*),

RUIQI GAO (*Department of Statistics, University of California, Los Angeles*),

TIAN HAN (*Department of Statistics, University of California, Los Angeles*),

AND

SONG-CHUN ZHU (*Department of Statistics, University of California, Los Angeles*)

Abstract.
The pattern theory of Grenander is a mathematical framework where patterns are represented by probability models on random variables of algebraic structures. In this paper, we review three families of probability models, namely, the discriminative models, the descriptive models, and the generative models. A discriminative model is in the form of a classifier. It specifies the conditional probability of the class label given the input signal. A descriptive model specifies the probability distribution of the signal, based on an energy function defined on the signal. A generative model assumes that the signal is generated by some latent variables via a transformation. We shall review these models within a common framework and explore their connections. We shall also review the recent developments that take advantage of the high approximation capacities of deep neural networks.

1. Introduction.

Initially developed by Grenander in the 1970s, the pattern theory [30, 31] is a unified mathematical framework for representing, learning and recognizing patterns that arise in science and engineering. The objects in pattern theory are usually of high complexity or dimensionality, defined in terms of the constituent elements and the bonds between them. The patterns of these objects are characterized by both the algebraic structures governed by local and global rules, as well as the probability distributions of the associated random variables. Such a framework encompasses most of the probability models in various disciplines. In the 1990s, Mumford [70] advocated the pattern theoretical framework for computer vision, so that learning and inference can be based on probability models.

Despite its generality, developing probability models in the pattern theoretical framework remains a challenging task. In this article, we shall review three families of models, which we

2000 *Mathematics Subject Classification.* Primary 62M40.

call the discriminative models, the descriptive models, and the generative models, following the terminology of [113]. A discriminative model is in the form of a classifier. It specifies the conditional probability of the output class label given the input signal. Such a model can be learned in the supervised setting where a training dataset of input signals and the corresponding output labels is provided. A descriptive model specifies the probability distribution of the signal, based on an energy function defined on the signal through some descriptive feature statistics extracted from the signal. Such models originated from statistical physics, where they are commonly called the Gibbs distributions [28]. The descriptive models belong to the broader class of energy-based models [59] that include non-probabilistic models as well as models with latent variables. A generative model assumes that the signal is generated by some latent variables via a deterministic transformation. A prototype example is factor analysis [83], where the signal is generated by some latent factors via a linear transformation. Both the descriptive models and generative models can be learned in the unsupervised setting where the training dataset only consists of input signals without the corresponding output labels.

In this paper, we shall review these three families of models within a common framework and explore their connections. We shall start from the flat linear forms of these models. Then we shall present the hierarchical non-linear models, where the non-linear mappings in these models are parametrized by neural networks [58, 56] that have proved exceedingly effective in approximating non-linear relationships.

Currently the most successful family of models are the discriminative models. A discriminative model is in the form of the conditional distribution of the class label given the input signal. The normalizing constant of such a probability model is a summation over the finite number of class labels or categories. It is readily available, so that the model can be easily learned from big datasets. The learning of the descriptive models and the generative models can be much more challenging. A descriptive model is defined as a probability distribution of the signal, which is usually of a high dimensionality. The normalizing constant of such a model is an integral over the high dimensional signal and is analytically intractable. A generative model involves latent variables that follow some prior distribution, so that the marginal distribution of the observed signal is obtained by integrating out the latent variables, and this integral is also analytically intractable. Due to the intractabilities of the integrals in the descriptive and generative models, the learning of such models usually requires Markov chain Monte Carlo (MCMC) sampling [25, 65]. Specifically, the learning of the descriptive models require MCMC sampling of the synthesized signals, while the learning of the generative models require MCMC sampling of the latent variables. Nonetheless, we shall show that such learning methods work reasonably well [106, 24, 35], where the gradient-based Langevin dynamics [71] can be employed conveniently for MCMC sampling, which is an inner loop within the gradient-based learning of the model parameters.

Because of the high capacity of the neural networks in approximating highly non-linear mappings, the boundary between representation and computation is blurred in neural networks. A deep neural network can be used to represent how the signal is generated or how the features are defined. It can also be used to approximate the solution of a computational problem such as optimization or sampling. For example, the iterative sampling of the latent variables of a generative model can be approximated by an inference model that provides the posterior samples directly, as is the case with the wake-sleep algorithm [41] and the variational auto-encoder

(VAE) [53, 78, 68]. As another example, the iterative sampling of a descriptive model can be approximated by a generative model that can generate the signal directly [104, 105]. In general, the solutions to the on-line computational problems can be encoded by high capacity neural networks, so that iterative computations only occur in the off-line learning of the model parameters.

The three families of models do not exist in isolation. There are intimate connections between them. In [33, 34], the authors proposed to integrate the descriptive and generative models into a hierarchical model. In [97, 98], the authors proposed data-driven MCMC where the MCMC is to fit the generative models, but the proposal distributions for MCMC transitions are provided by discriminative models. The discriminative model and the descriptive model can be translated into each other via the Bayes rule. Tu [96] exploited this relationship to learn the descriptive model via discriminative training, thus unifying the two models. Similarly, the discriminative model can be paired with the generative model in the generative adversarial networks (GAN) [29], and the adversarial learning has become an alternative framework to likelihood-based learning. The descriptive model and the generative model can also be paired up so that they can jumpstart each other's MCMC sampling [104, 105]. Moreover, the family of descriptive models and the family of generative models overlap in terms of undirected latent energy-based models [59].

2. Non-hierarchical linear forms of the three families. We shall first review the non-hierarchical linear forms of the three families of models within a common framework.

2.1. *Discriminative models.* This subsection reviews the linear form of the discriminative models.

The table below displays the dataset for training the discriminative models:

	input	features	output
1	$X_1^\top$	$h_1^\top$	Y_1
2	$X_2^\top$	$h_2^\top$	Y_2
...			
n	$X_n^\top$	$h_n^\top$	Y_n

There are n training examples. For the i-th example, let $X_i = (x_{ij}, j = 1, ..., p)^\top$ be the p-dimensional input signal (the (n, p) notation is commonly used in statistics to denote the number of observations and the number of predictors respectively). Let Y_i be the outcome label. In the case of classification, Y_i is categorical or binary. $h_i = (h_{ik}, k = 1, ..., d)^\top$ is the d-dimensional vector of features or hidden variables.

The discriminative models can be represented by the diagram below:

$$\begin{array}{rc} \text{output}: & Y_i \\ & \uparrow \\ \text{features}: & h_i \\ & \uparrow \\ \text{input}: & X_i \end{array} \tag{2.1}$$

where the vector of features h_i is computed from X_i via $h_i = h(X_i)$. In a non-hierarchical or flat model, the feature vector h_i is designed, not learned, i.e., $h()$ is a pre-specified non-linear transformation.

For the case of binary classification where $Y_i \in \{+1, -1\}$, (Y_i, X_i) follow a logistic regression

$$\log \frac{\Pr(Y_i = +|X_i)}{\Pr(Y_i = -|X_i)} = h_i^\top \theta + b, \tag{2.2}$$

where θ is the d-dimensional vector of weight or coefficient parameters, and b is the bias or intercept parameter. The classification can also be based on the perceptron model

$$\hat{Y}_i = \text{sign}(h_i^\top \theta + b), \tag{2.3}$$

where $\text{sign}(r) = +1$ if $r \geq 0$, and $\text{sign}(r) = -1$ otherwise. Both the logistic regression and the perceptron can be generalized to the multi-category case. The bias term b can be absorbed into the weight parameters θ if we fix $h_{i1} = 1$.

Let $f(X) = h(X)^\top \theta$. $f(X)$ captures the relationship between X and Y. Because $h(X)$ is non-linear, $f(X)$ is also non-linear. We say the model is in the linear form because it is linear in θ, or $f(X)$ is a linear combination of the features in $h(X)$. The following are the choices of $h()$ in various discriminative models.

Kernel machine [12]: $h_i = h(X_i)$ is implicit, and the dimension of h_i can potentially be infinite. The implementation of this method is based on the kernel trick $\langle h(X), h(X') \rangle = K(X, X')$, where K is a kernel that is explicitly used by the classifier such as the support vector machine [12]. $f(X) = h(X)^\top \theta$ belongs to the reproducing kernel Hilbert space where the norm of f can be defined as the Euclidean norm of θ, and the norm is used to regularize the model. A Bayesian treatment leads to the Gaussian process, where θ is assumed to follow $\text{N}(0, \sigma^2 I_d)$, and I_d is the identity matrix of dimension d. $f(X)$ is a Gaussian process with $\text{Cov}(f(X), f(X')) = \sigma^2 K(X, X')$.

Boosting machine [22]: For $h_i = (h_{ik}, k = 1, ..., d)^\top$, each $h_{ik} \in \{+, -\}$ is a weak classifier or a binary feature extracted from X, and $f(X) = h(X)^\top \theta$ is a committee of weak classifiers.

CART [6]: In the classification and regression trees, there are d rectangle regions $\{R_k, k = 1, ..., d\}$ resulted from recursive binary partition of the space of X, and each $h_{ik} = 1(X_i \in R_k)$ is the binary indicator such that $h_{ik} = 1$ if $X_i \in R_k$ and $h_{ik} = 0$ otherwise. $f(X) = h(X)^\top \theta$ is a piecewise constant function.

MARS [23]: In the multivariate adaptive regression splines, the components of $h(X)$ are hinge functions such as $\max(0, x_j - t)$ (where x_j is the j-th component of X, $j = 1, ..., p$, and t is a threshold) and their products. It can be considered a continuous version of CART.

Encoder and decoder: In the diagram in (2.1), the transformation $X_i \to h_i$ is called an encoder, and the transformation $h_i \to Y_i$ is called a decoder. In the non-hierarchical model, the encoder is designed, and only the decoder is learned.

The outcome Y_i can also be continuous or a high-dimensional vector. The learning then becomes a regression problem. Both classification and regression are about supervised learning because for each input X_i, an output Y_i is provided as supervision. The reinforcement learning is similar to supervised learning except that the guidance is in the form of a reward function.

2.2. *Descriptive models.*

This subsection describes the linear form of the descriptive models and the maximum likelihood learning algorithm.

The descriptive models [113] can be learned in the unsupervised setting, where Y_i are not observed, as illustrated by the table below:

	input	features	output
1	$X_1^\top$	$h_1^\top$	?
2	$X_2^\top$	$h_2^\top$	?
...			
n	$X_n^\top$	$h_n^\top$	?

The linear form of the descriptive model is an exponential family model. It specifies a probability distribution on the signal X_i via an energy function that is a linear combination of the features,

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[h(X)^\top \theta\right] p_0(X), \tag{2.4}$$

where $h(X)$ is the d-dimensional feature vector extracted from X, and θ is the d-dimensional vector of weight parameters. $p_0(X)$ is a known reference distribution such as the white noise model $X \sim \mathrm{N}(0, \sigma^2 I_p)$, or the uniform distribution within a bounded range.

$$Z(\theta) = \int \exp[h(X)^\top \theta] p_0(X) dX = \mathrm{E}_{p_0}\{\exp[h(X)^\top \theta]\} \tag{2.5}$$

is the normalizing constant (E_p denotes the expectation with respect to p). It is analytically intractable.

The descriptive model (2.4) has the following information theoretical property [16, 117, 2]. Let $P_{\rm data}$ be the distribution that generates the training data $\{X_i\}$. Let $\Theta = \{p_\theta, \forall \theta\}$ be the family of distributions defined by the descriptive model. Let $\Omega = \{p : \mathrm{E}_p[h(X)] = \hat{h}\}$, where $\hat{h} = \mathrm{E}_{P_{\rm data}}[h(X)]$. $\hat{h}$ can be estimated from the observed data by the sample average $\sum_{i=1}^n h(X_i)/n$. Ω is the family of distributions that reproduce the observed $\hat{h}$. Let $\hat{p} = p_{\hat{\theta}} \in$

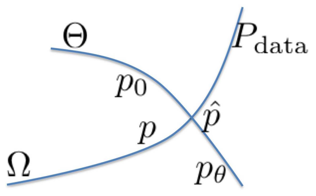


FIG. 1. The two curves illustrate Θ and Ω respectively, where each point is a probability distribution.

$\Theta \cup \Omega$ be the intersection between Θ and Ω. Then for any $p_\theta \in \Theta$ and any $p \in \Omega$, we have $\mathrm{KL}(p\|p_\theta) = \mathrm{KL}(p\|\hat{p}) + \mathrm{KL}(\hat{p}\|p_\theta)$, which can be interpreted as a Pythagorean property that defines orthogonality. $\mathrm{KL}(p\|q) = \mathrm{E}_p[\log(p(X)/q(X))]$ denotes the Kullback-Leibler divergence from p to q. Thus Θ and Ω are orthogonal to each other, $\Theta \perp \Omega$, as illustrated by Figure 1.

This leads to the following dual properties of $\hat{p}$, which can be considered the learned model:

(1) Maximum likelihood. $\hat{p} = \arg\min_\Theta \mathrm{KL}(P_{\rm data}\|p_\theta)$. That is, $\hat{p}$ is the projection of $P_{\rm data}$ on Θ. $\mathrm{KL}(P_{\rm data}\|p_\theta) = \mathrm{E}_{P_{\rm data}}[\log P_{\rm data}(X)] - \mathrm{E}_{P_{\rm data}}[\log p_\theta(X)]$. The second term $\mathrm{E}_{P_{\rm data}}[\log p_\theta(X)]$ is the population version of the log-likelihood. Thus minimizing $\mathrm{KL}(P_{\rm data}\|p_\theta)$ is equivalent to maximizing the likelihood.

(2) Maximum entropy: $\hat{p} = \arg\min_{\Omega} \mathrm{KL}(p\|p_0)$. That is, $\hat{p}$ is the minimal modification of p_0 to reproduce the observed feature statistics $\hat{h}$. $\mathrm{KL}(p\|p_0) = \mathrm{E}_p[\log p(X)] - \mathrm{E}_p[\log p_0(X)]$. If p_0 is the uniform distribution, then the second term is a constant, and the first term is the negative entropy. In that case, minimizing $\mathrm{KL}(p\|p_0)$ is equivalent to maximizing the entropy over Ω.

Given the training data $\{X_i\}$, let $\mathcal{L}(\theta) = \sum_{i=1}^{n} \log p_\theta(X_i)/n$ be the log-likelihood. The gradient of $\mathcal{L}(\theta)$ is

$$\mathcal{L}'(\theta) = \frac{1}{n}\sum_{i=1}^{n} h(X_i) - \mathrm{E}_\theta[h(X)], \tag{2.6}$$

because $\partial \log Z(\theta)/\partial\theta = \mathrm{E}_\theta[h(X)]$, where E_θ denotes the expectation with respect to p_θ. This leads to a stochastic gradient ascent algorithm for maximizing $\mathcal{L}(\theta)$,

$$\theta_{t+1} = \theta_t + \eta_t \left[\frac{1}{n}\sum_{i=1}^{n} h(X_i) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} h(\tilde{X}_i)\right], \tag{2.7}$$

where $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$ are random samples from p_{θ_t}, and η_t is the learning rate. The learning algorithm has an "analysis by synthesis" interpretation. The $\{\tilde{X}_i\}$ are the synthesized data generated by the current model. The learning algorithm updates the parameters in order to make the synthesized data similar to the observed data in terms of the feature statistics. At the maximum likelihood estimate $\hat{\theta}$, the model matches the data: $\mathrm{E}_{\hat{\theta}}[h(X)] = \mathrm{E}_{P_{\rm data}}[h(X)]$.

One important class of descriptive models are the Markov random field models [5, 26], such as the Ising model in statistical physics. Such models play an important role in the pattern theory.

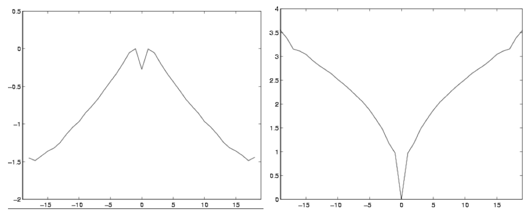


FIG. 2. Two types of potential functions learned by [115] from natural images. The function on the left encourages big filter responses and creates patterns via reaction, while the function on the right prefers small filter responses and smoothes out the synthesized image via diffusion.

One example of the descriptive model (2.4) is the FRAME (Filters, Random field, And Maximum Entropy) model [117, 103], where $h(X)$ consists of histograms of responses from a bank of filters. In a simplified non-convolutional version, $h(X)^\top\theta = f(WX) = \sum_{k=1}^{d} f_k(W_k X)$, where W is a $d \times p$ matrix, and W_k is the k-th row of W. WX consists of the d filter responses with each row of W being a linear filter. $(f_k, k = 1, ..., d)$ are d one-dimensional potential functions applied respectively to the d elements of WX. In the FRAME model, the rows of W are a bank of Gabor wavelets or filters [15]. Given the filters, [115] learned the potential functions $(-f_k, k = 1, ..., d)$ from natural images. There are two types of potential functions as shown in Figure 2 taken from [115]. The function on the left encourages big filter responses while the function on the right prefers small filter responses. [115] used the Langevin dynamics to sample from the learned model. The gradient descent component of the dynamics is interpreted as the

Gibbs Reaction And Diffusion Equations (GRADE), where the function on the left of Figure 2 is for reaction to create patterns, while the function on the right is for diffusion to smooth out the synthesized image.

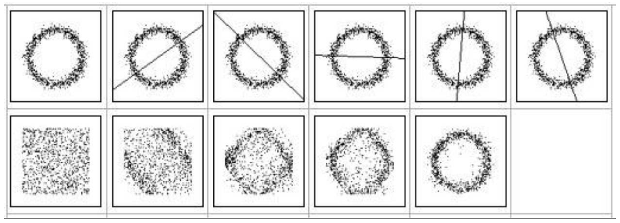


FIG. 3. Learning a two dimensional FRAME model by sequentially adding rows to W [64]. Each row of W corresponds to a projection of the data. Each step finds the projection that reveals the maximum difference between the observed data and the synthesized data generated by the current model.

In [64], the authors illustrated the idea of learning $W = (W_k, k = 1, ..., d)$ by a two-dimensional example. Each step of the learning algorithm adds a row W_k to the current W. Each row corresponds to a projection of X. Each step finds a direction of the projection that reveals the maximum difference between the data points sampled from the current model and the observed data points. The learning algorithm then updates the model to match the marginal distributions of the model and the data in that direction. After a few steps, the distribution of the learned model is almost the same as the distribution of the observed data. See Figure 3 for an illustration. By assuming a parametric differentiable form for $f_k()$, W can be learned by gradient descent. Such models are called product of experts [40, 93] or field of experts [80].

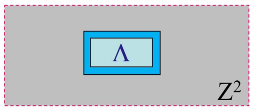


FIG. 4. Under the uniform distribution of images defined on a large lattice (that goes to $\mathbf{Z}^2$) where the images share the same marginal histograms of filter responses, the conditional distribution of the local image patch given its boundary (in blue color) follows the FRAME model.

The FRAME model is convolutional, where the rows of W can be partitioned into different groups, and the rows in the same group are spatially translated versions of each other, like wavelets. They are called filters or kernels. The model can be justified by a uniform distribution over the images defined on a large lattice that goes to $\mathbf{Z}^2$, where all the images share the same marginal histograms of filter responses. Under such a uniform distribution, the distribution of the local image patch defined on a local lattice Λ conditional on its boundary (illustrated by the blue color, including all the pixels outside Λ that can be covered by the same filters as the pixels within Λ) follows the FRAME model [103]. See Figure 4 for an illustration.

2.3. *Generative models.* This subsection reviews various versions of the linear generative models. These models share the same linear form, but they differ in terms of the prior assumptions of the latent factors or coefficients.

Like the descriptive models, the generative models can be learned in the unsupervised setting, where Y_i are not observed, as illustrated below:

	input	hidden	output
1	$X_1^\top$	$h_1^\top$	?
2	$X_2^\top$	$h_2^\top$	?
...			
n	$X_n^\top$	$h_n^\top$	?

In a generative model, the vector h_i is not a vector of features extracted from the signal X_i. h_i is a vector of hidden variables that is used to generate X_i, as illustrated by the following diagram:

$$\begin{array}{rc} \text{hidden}: & h_i \\ & \downarrow \\ \text{input}: & X_i \end{array} \quad (2.8)$$

The components of the d-dimensional h_i are variably called factors, sources, components or causes.

Auto-encoder: h_i is also called a code in the auto-encoder illustrated by the following diagram:

$$\begin{array}{rc} \text{code}: & h_i \\ & \uparrow\downarrow \\ \text{input}: & X_i \end{array} \quad (2.9)$$

The direction from h_i to X_i is called the decoder, and the direction from X_i to h_i is called the encoder. The decoder corresponds to the generative model in (2.8), while the encoder can be considered the inference model.

Distributed representation and disentanglement: $h_i = (h_{ik}, k = 1, ..., d)$ is called a distributed representation of X_i. Usually the components of h_i, $(h_{ik}, k = 1, ..., d)$, are assumed to be independent, and (h_{ik}) are said to disentangle the variations in X_i.

Embedding: h_i can also be considered the coordinates of X_i, if we embed X_i into a low-dimensional space, as illustrated by the following diagram:

$$\begin{array}{c} \leftarrow h_i \rightarrow \\ | \\ \leftarrow X_i \rightarrow \end{array} \quad (2.10)$$

In the training data, we find a h_i for each X_i, so that $\{h_i, i = 1, ..., n\}$ preserve the relative relations between $\{X_i, i = 1, ..., n\}$. The prototype example of embedding is multi-dimensional scaling, where we want to preserve the Euclidean distances between the examples. A more recent example of embedding is local linear embedding [81]. In the embedding framework, there are no explicit encoder and decoder.

Linear generative model: The linear form of the generative model is as follows:

$$X_i = Wh_i + \epsilon_i, \quad (2.11)$$

for $i = 1, ..., n$, where W is a $p \times d$ dimensional matrix (p is the dimensionality of X_i and d is the dimensionality of h_i), and ϵ_i is a p-dimensional residual vector. The following are the interpretations of W:

(1) Loading matrix: Let $W = (w_{jk})_{p\times d}$. $x_{ij} \approx \sum_{k=1}^{d} w_{jk}h_{ik}$, i.e., each component of X_i, x_{ij}, is a linear combination of the latent factors. w_{jk} is the loading weight of factor k on variable j.

(2) Basis vectors: Let $W = (W_k, k = 1, ..., d)$, where W_k is the k-th column of W. $X_i \approx \sum_{k=1}^{d} h_{ik}W_k$, i.e., X_i is a linear superposition of the basis vectors (W_k), where h_{ik} are the coefficients.

(3) Matrix factorization: $(X_1, ..., X_n) \approx W(h_1, ..., h_n)$, where the $p \times n$ matrix $(X_1, ..., X_n)$ is factorized into the $p \times d$ matrix W and the $d \times n$ matrix $(h_1, ..., h_n)$.

The following are some of the commonly assumed prior distributions or constraints on h_i.

Factor analysis [83]: $h_i \sim \mathrm{N}(0, I_d)$, $X_i = Wh_i + \epsilon_i$, $\epsilon_i \sim \mathrm{N}(0, \sigma^2 I_p)$, and ϵ_i is independent of h_i. The dimensionality of h_i, which is d, is smaller than the dimensionality of X_i, which is p. The factor analysis is very similar to the principal component analysis (PCA), which is a popular tool for dimension reduction. The difference is that in factor analysis, the column vectors of W do not need to be orthogonal to each other.

The factor analysis model originated from psychology, where X_i consists of the test scores of student i on p subjects. h_i consists of the verbal intelligence and the analytical intelligence of student i ($d = 2$). Another example is the decathlon competition, where X_i consists of the scores of athlete i on $p = 10$ sports, and h_i consists of athlete i's speed, strength and endurance ($d = 3$).

Independent component analysis [48]: In ICA, for $h_i = (h_{ik}, k = 1, ..., d)$, $h_{ik} \sim P_k$ independently, and P_k are assumed to be heavy-tailed distributions. For analytical tractability, ICA assumes that $d = p$, and $\epsilon_i = 0$. Hence $X_i = Wh_i$, where W is a squared matrix assumed to be invertible, so that $h_i = AX_i$, where $A = W^{-1}$. Let $P(h_i) = \prod_{k=1}^{d} P_k(h_{ik})$. The marginal distribution of X_i has a closed form $X_i \sim P(AX)|\det(A)|$. The ICA model is both a generative model and a descriptive model.

Sparse coding [73]: In the sparse coding model, the dimensionality of h_i, which is d, is bigger than the dimensionality of X_i, which is p. However, $h_i = (h_{ik}, k = 1, ..., d)$ is a sparse vector, meaning that only a small number of h_{ik} are non-zero, although for different example i, the non-zero elements in h_i can be different. Thus unlike PCA, sparse coding provides adaptive dimension reduction. $W = (W_k, k = 1, ..., d)$ is called a redundant dictionary because $d > p$, and each W_k is a basis vector or a "word" in the dictionary. Each $X_i \approx Wh_i = \sum_{k=1}^{d} h_{ik}W_k$ is explained by a small number of W_k selected from the dictionary, depending on which h_{ik} are non-zero. The inference of the sparse vector h_i can be accomplished by Lasso or basis pursuit [94, 8] that minimizes $\sum_{i=1}^{n} \left[\|X_i - Wh_i\|^2 + \lambda\|h_i\|_{\ell_1}\right]$, which imposes the sparsity inducing ℓ_1 regularization on h_i with a regularization parameter λ.

A Bayesian probabilistic formulation is to assume a spike-slab prior: $h_{ik} \sim \rho\delta_0 + (1 - \rho)\mathrm{N}(0, \tau^2)$ with a small $1 - \rho$, which is the probability that h_{ik} is non-zero.

Figure 5 displays a sparse code learned from a training set of natural image patches of size 12×12 [73]. Each column of W, W_k, is a basis vector that can be made into an image patch as shown in the figure.

Non-negative matrix factorization [60]: In NMF, h_i is constrained to have non-negative components, i.e., $h_{ik} \geq 0$ for all k. It is also called positive factor analysis [75]. The rationale for NMF is that the parts of a pattern should be additive and the parts should contribute positively.

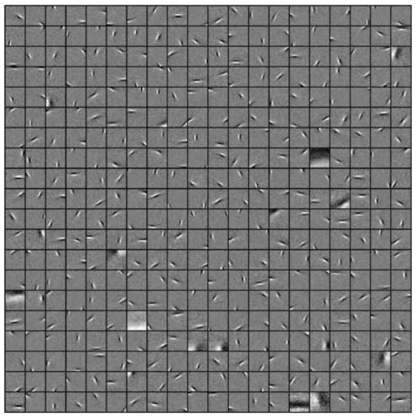


FIG. 5. Sparse coding [73]: learned basis vectors from natural image patches. Each image patch in the picture is a column vector of W.

Matrix factorization for recommender system [54]: In recommender system, $X_i = (x_{ij}, j = 1, ..., p)$ are the ratings of user i on the p items. For instance, in the Netflix example, there are n users and p movies, and x_{ij} is user i's rating of movie j. Let w_j be the j-th row of matrix W, then $x_{ij} = \langle w_j, h_i \rangle + \epsilon_{ij}$, where h_i characterizes the desires of user i in d aspects, and w_j characterizes the desirabilities of item j in the corresponding aspects. The rating matrix $(X_i, i = 1, ..., n)$ thus admits a rank d factorization. The rating matrix is in general incomplete. However, we can still estimate (h_i) and (w_j) from the observed ratings and use them to complete the rating matrix for the purpose of recommendation.

Probabilistic formulation: In the above models, there is a prior model $h_i \sim p(h)$ or a prior constraint such as h_i is sparse or non-negative. There is a linear generative model $X_i = Wh_i + \epsilon_i$, with $\epsilon_i \sim \mathrm{N}(0, \sigma^2 I_p)$, for $i = 1, ..., n$. This defines the conditional distribution $p(X|h; W)$. The joint distribution is $p(h)p(X|h; W) = p(h, X|W)$. The marginal distribution is obtained by integrating out h:

$$p(X|W) = \int p(h)p(X|h; W)dh = \int p(h, X|W)dh. \tag{2.12}$$

This integral is analytically intractable. According to the Bayes rule, h can be inferred from X based on the posterior distribution, $p(h|X; W) = p(h, X|W)/p(X|W)$, which is proportional to $p(h, X|W)$ as a function of h. We call $p(h|X; W)$ the inference model.

In the auto-encoder terminology, $p(h)$ and $p(X|h; W)$ define the decoder, $p(h|X; W)$ defines the encoder. In factor analysis and independent component analysis, h can be inferred in closed form. For other models, however, h needs to be inferred by an iterative algorithm.

Restricted Boltzmann machine [42]: In RBM, unlike the above models, there is no explicit prior $p(h)$. The model is defined by the joint distribution

$$\begin{aligned}(h_i, X_i) \sim p(h, X|W) &= \frac{1}{Z(W)} \exp\left[\sum_{j,k} w_{jk} x_j h_k\right] &(2.13)\\ &= \frac{1}{Z(W)} \exp\left[X^\top W h\right]. &(2.14)\end{aligned}$$

The above model assumes that both h_i and X_i are binary. Under the above model, both the generative distribution $p(X|h;W)$ and the inference distribution $p(h|X;W)$ are independent logistic regressions. We may modify the model slightly to make X continuous, so that in the modified model, the generative distribution $p(X|h;W)$ is normal linear regression: $X = Wh + \epsilon$, with $\epsilon \sim \mathrm{N}(0, \sigma^2 I_p)$. The inference model, $p(h|X;W)$, is logistic regression, $h \sim \mathrm{logistic}(W^\top X)$, i.e., $\Pr(h_k = 1|X;W) = \mathrm{sigmoid}(\sum_{j=1}^p w_{jk}x_j)$, where $\mathrm{sigmoid}(r) = 1/(1+e^{-r})$.

If we sum out h, the marginal distribution $p(X|W) = \sum_h p(h, X|W)$ can be obtained in closed form, and $p(X|W)$ is a descriptive model.

RBM-like auto-encoder [100, 4]: The RBM leads to the following auto-encoder: Encoder: $h_k = \mathrm{sigmoid}(\sum_{j=1}^p w_{jk}x_j)$, i.e., $h = \mathrm{sigmoid}(W^\top X)$; Decoder: $X = Wh$.

Like the descriptive model, the generative model can also be learned by maximum likelihood. However, unlike the "analysis by synthesis" scheme for learning the descriptive model, the learning algorithm for generative model follows an "analysis by inference" scheme. Within each iteration of the learning algorithm, there is an inner loop for inferring h_i for each X_i. The most rigorous inference method is to sample h_i from the posterior distribution or the inference distribution $p(h_i|X_i;W)$. After inferring h_i for each X_i, we can then update the model parameters by analyzing the "imputed" dataset $\{(h_i;X_i)\}$, by fitting the generative distribution $p(X|h;W)$. The EM algorithm [17] is an example of this learning scheme, where the inference step is to compute expectation with respect to $p(h_i|X_i;W)$. From a Monte Carlo perspective, it means we make multiple imputations [82] or make multiple guesses of h_i to account for the uncertainties in $p(h_i|X_i;W)$. Then we analyze the multiply imputed dataset to update the model parameters.

3. Interactions between different families.

3.1. *Discriminative learning of descriptive model.* This subsection shows that the descriptive model can be learned discriminatively.

The descriptive model (2.4) can be connected to the discriminative model (2.2) if we treat $p_0(X)$ as the distribution of the negative examples, and $p_\theta(X)$ as the distribution of the positive examples. Suppose we generate the data as follows: $Y_i \sim \mathrm{Bernoulli}(\rho)$, i.e., $\Pr(Y_i = 1) = \rho$, which is the prior probability of positive examples. $[X_i \mid Y_i = 1] \sim p_\theta(X)$, and $[X_i \mid Y_i = 0] \sim p_0(X)$. According to the Bayes rule

$$\log \frac{\Pr(Y_i = 1 \mid X_i)}{\Pr(Y_i = 0 \mid X_i)} = h(X_i)^\top \theta - \log Z(\theta) + \log[\rho/(1-\rho)], \tag{3.1}$$

which corresponds to (2.2) with $b = -\log Z(\theta) + \log[\rho/(1-\rho)]$.

Tu [96] made use of this fact to estimate p_θ discriminatively. The learning algorithm starts from p_0. At step t, we let the current p_t serve as the negative distribution, and generate synthesized examples from p_t. Then we fit a logistic regression by treating the examples generated by p_t as the negative examples, and the observed examples as the positive examples. Let θ be the estimated parameter of this logistic regression. We then let $p_{t+1}(X) = \exp(h(X)^\top\theta)p_t(X)/Z$. See [96] for an analysis of the convergence of the learning algorithm.

Figure 6 taken from [96] illustrates the learning process by starting from the uniform p_0. By iteratively fitting the logistic regression and modifying the distribution, the learned distribution converges to the true distribution.

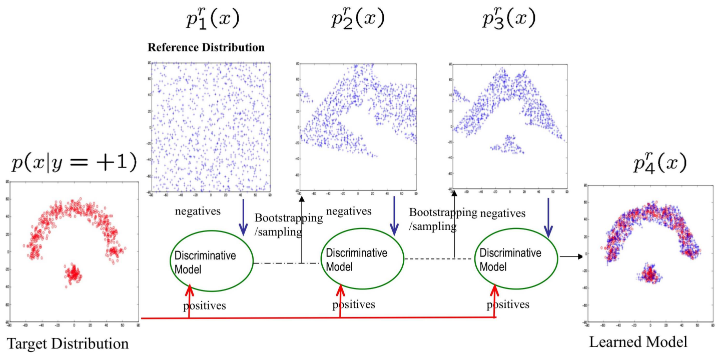


FIG. 6. Discriminative learning of the descriptive model [96]. By fitting a logistic regression to discriminate between the observed examples and the synthesized examples generated by the current model, we can modify the current model according to the fitted logistic regression, so that the modified model gets closer to the distribution of the observed data.

3.2. *Integration of descriptive and generative models.* Natural images contain both stochastic textures and geometric objects (as well as their parts). The stochastic textures can be described by some feature statistics pooled over the spatial domain, while the geometric objects can be represented by image primitives or textons. The psychophysicist Julesz [51] studied both texture

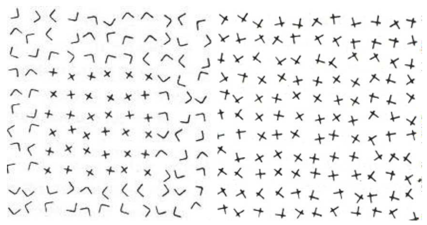


FIG. 7. Pre-attentive vision is sensitive to local patterns called textons.

statistics and textons. He conjectured that pre-attentive human vision is sensitive to local patterns called textons. Figure 7 illustrates the basic idea. Inspired by Julesz's work, in [114], the authors

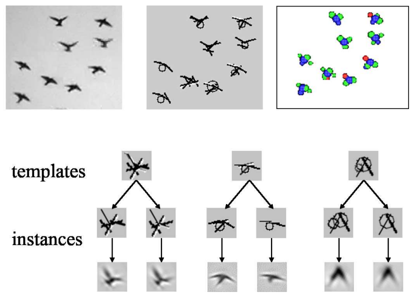


FIG. 8. A model of textons [114], where each texton is a composition of a small number of wavelets.

proposed a generative model for textons, where each texton is a composition of a small number of wavelets, as illustrated by Figure 8. The model is a generalization of the sparse coding model of [73].

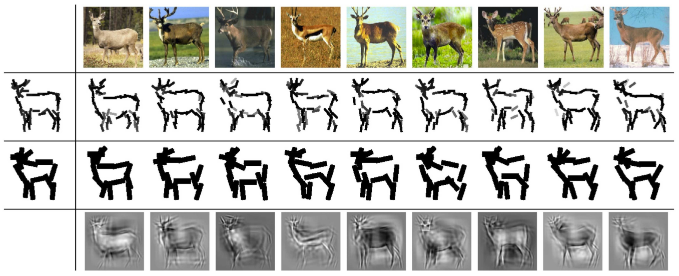


FIG. 9. Active basis model [102, 44]: each active basis template is a composition of wavelets selected from a dictionary, and the wavelets are allowed to shift their locations and orientations to account for shape deformation. Here each wavelet is illustrated by a bar. The templates are learned at two different scales. The observed images can be reconstructed by the wavelets of the deformed templates.

Building on the texton model of [114], [102, 44] proposed an active basis model, where each model is a composition of wavelets selected from a dictionary, and the wavelets are allowed to shift their locations and orientations to account for shape deformation. See Figure 9 for an illustration.

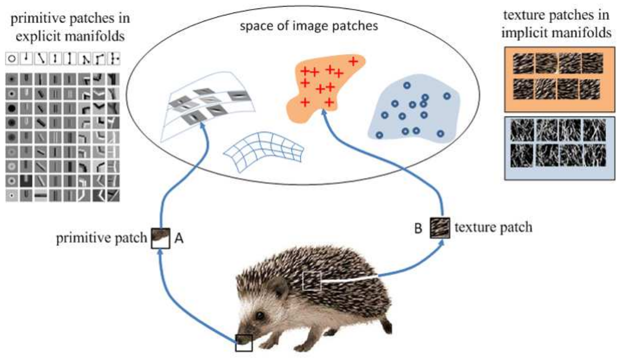

primitive patches in explicit manifolds
space of image patches
texture patches in implicit manifolds
primitive patch
A
B
texture patch

FIG. 10. Hybrid image template [87]: integrating generative model for shape template and the descriptive model for texture.

The texton model and the active basis model are generative models. However, they do not account for stochastic texture patterns. [87] proposed to integrate the generative model for shape templates and the descriptive model for stochastic textures, as illustrated by Figure 10. A similar model was developed by [34] to model both the geometric structures and stochastic textures by generative models and descriptive models respectively.

In [33], the authors provided another integration of the generative model and the descriptive model, where the lowest layer is a generative model such as the wavelet sparse coding model [73], but the spatial distribution of the wavelets is governed by a descriptive model.

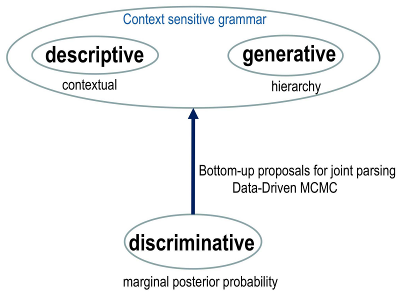

Context sensitive grammar
descriptive
contextual
generative
hierarchy
Bottom-up proposals for joint parsing
Data-Driven MCMC
discriminative
marginal posterior probability

FIG. 11. Data-driven MCMC: when fitting the generative models and descriptive models using MCMC, the discriminative models can be employed to provide proposals for MCMC transitions.

3.3. *DDMCMC: integration of discriminative and generative models.* In [97, 98], the authors proposed a data-driven MCMC method for fitting the generative models as well as the descriptive models to the data. Fitting such models usually require time-consuming MCMC. In [97, 98], the authors proposed to speed up the MCMC by using the discriminative models to provide the proposals for the Metropolis-Hastings algorithm. See Figure 11 for an illustration.

4. Hierarchical forms of the three families.

This section presents the hierarchical non-linear forms of the three families of models, where the non-linear mappings are parametrized by neural networks, in particular, the convolutional neural networks.

4.1. *Recent developments.* During the past few years, deep convolutional neural networks (CNNs or ConvNets) [58, 56] and recurrent neural networks (RNNs) [43] have transformed the fields of computer vision, speech recognition, natural language processing, and other fields in artificial intelligence (AI). Even though these neural networks were invented decades ago, their potentials were realized only recently mainly because of the following two factors. (1) The availability of big training datasets such as Imagenet [18]. (2) The improvement in computing power, mainly brought by the graphical processing units (GPUs). These two factors, together with some recent clever tweaks and inventions such as rectified linear units [56], batch normalization [49], residual networks [39], etc., enable the training of very deep networks (e.g., 152 layers with 60 million parameters in a residual network for object recognition [39]) that achieve impressive performances on many tasks in AI (a recent example being Alpha Go Zero [89]).

One key reason for the successes of deep neural networks is that they are universal and flexible function approximators. For instance, a feedforward neural network with rectified linear units is a piecewise linear function with recursively partitioned linear pieces that can approximate any continuous non-linear mapping [69]. However, this does not fully explain the "unreasonable effectiveness" of deep neural networks. The stochastic gradient descent algorithm that is commonly employed to train the neural networks is expected to approach only a local minimum of the highly non-convex objective function. However, for large and deep networks, it appears that most of the local modes are equally good [10] in terms of training and testing errors, and the apparent vices of local modes and stochasticity in the mini-batch on-line training algorithm actually turn out to be big virtues in that they seem to prevent overfitting and lead to good generalization [10].

The approximation capacities of the deep neural networks have been extensively exploited in supervised learning (such as classification networks and regression networks) and reinforcement learning (such as policy networks and value networks). They have also proven to be useful for unsupervised learning and generative modeling, where the goal is to learn features or hidden variables from the observed signals without external guidance such as class labels or rewards. The unsupervised learning is often accomplished in the context of a generative model (or an auto-encoder), which explains or characterizes the observed examples.

4.2. *Discriminative models by convolutional neural networks.* The neural networks in general and the convolutional neural networks (ConvNet or CNN) in particular were initially designed for discriminative models. Let X be the p-dimensional input vector, and Y be the output. We want to predict Y by $\hat{Y}$ which is a non-linear transformation of X: $\hat{Y} = f_\theta(X)$, where f is parametrized by parameters θ. In a feedforward neural network, f is a composition of L layers of liner mappings followed by coordinate-wise non-linear rectifications, as illustrated by the following diagram:

$$X \to h^{(1)} \to ...h^{(l-1)} \to h^{(l)} \to ... \to h^{(L)} \to \hat{Y}, \tag{4.1}$$

where $h^{(l)}$ is a $d^{(l)}$ dimensional vector which is defined recursively by

$$h^{(l)} = f^{(l)}(W^{(l)}h^{(l-1)} + b^{(l)}), \tag{4.2}$$

for $l = 1, ..., L$. We may treat X as $h^{(0)}$, and $\hat{Y}$ as $h^{(L+1)}$ and $\theta = (W^{(l)}, b^{(l)}, l = 1, ..., L+1)$. $W^{(l)}$ is the weight matrix and $b^{(l)}$ is the bias or intercept vector at layer l. $f^{(l)}$ is coordinate-wise transformation, i.e., for a vector $v = (v_1, ..., v_d)^\top$, $f^{(l)}(v) = (f^{(l)}(v_1), ..., f^{(l)}(v_d))^\top$.

Compared to the discriminative models in the previous section, we now have multiple layers of features $(h^{(l)}, l = 1, ..., L)$. They are recursively defined via (4.2), and they are to be learned from the training data instead of being designed.

For classification, suppose there are K categories, the conditional probability of category k given input X is given by the following soft-max probability:

$$\Pr(Y = k \mid X) = \frac{\exp(f_{\theta_k}(X))}{\sum_{k=1}^{K} \exp(f_{\theta_k}(X))}, \tag{4.3}$$

where $f_{\theta_k}(X)$ is the score for category k. We may take $f_{\theta_k}(X) = h^{(L)\top}W_k^{(L+1)} + b_k^{(L+1)}$. This final classification layer is usually called the soft-max layer.

The most commonly used non-linear rectification in modern neural nets is the Rectified Linear Unit (ReLU) [56]: $f^{(l)}(a) = \max(0, a)$. The resulting function $f_\theta(X)$ can be considered a multi-dimensional linear spline, i.e., a piecewise linear function. Recall a one-dimensional linear spline is of the form $f(x) = b + \sum_{k=1}^{d} w_k \max(0, x - a_k)$, where a_k are the knots. At each knot a_k, the linear spline takes a turn and changes its slope by w_k. With enough knots, $f(x)$ can approximate any non-linear continuous function. We can view this $f(x)$ as a simplified two-layer network, with $h_k = \max(0, x - a_k)$. The basis function $\max(0, x - a_k)$ is two-piece linear function with a bending at a_k. For multi-dimensional input X, a two-layer network with one-dimensional output is of the following form $f(X) = b^{(2)} + \sum_{k=1}^{d} W_k^{(2)} h_k^{(1)}$, where $h_k^{(1)} = \max(0, W_k^{(1)} X + b_k^{(1)})$, and $W_k^{(1)}$ is the k-th row of $W^{(1)}$. The basis function $\max(0, W_k^{(1)} X + b_k^{(1)})$ is again a two-piece linear function with a bending along the line $W_k^{(1)} X + b_k^{(1)} = 0$. The dividing lines $\{W_k^{(1)} X + b_k^{(1)} = 0, k = 1, ..., d^{(1)}\}$ partition the domain of X into up to $2^{d^{(1)}}$ pieces, and $f(X)$ is a continuous piecewise linear function over these pieces.

In the multi-layer network, the hierarchical layers of $\{h^{(l)}, l = 1, ..., L\}$ partition the domain of X recursively, creating a piecewise linear function with exponentially many pieces [76]. Such reasoning also applies to other forms of rectification functions $f^{(l)}$, as long as they are non-linear and create bending. This makes the neural network an extremely powerful machine for function approximation and interpolation. The recursive partition in neural nets is similar to CART and MARS, but is more flexible.

Back-propagation. Both $\partial f_\theta(X)/\partial \theta$ and $\partial f_\theta(X)/\partial X$ can be computed by the chain-rule back-propagation, and they share the computation of $\partial h^{(l)}/\partial h^{(l-1)} = f^{(l)\prime}(W^{(l)} h^{(l-1)} + b^{(l)}) W^{(l)}$ in the chain rule. Because $f^{(l)}$ is coordinate-wise, $f^{(l)\prime}$ is a diagonal matrix.

A recent invention [39] is to reparametrize the mapping (4.2) by $h^{(l)} = h^{(l-1)} + f^{(l)}(W^{(l)} h^{(l-1)} + b^{(l)})$, where $f^{(l)}(W^{(l)} h^{(l-1)} + b^{(l)})$ is used to model the residual term. This enables the learning of very deep networks. One may think of it as modeling an iterative algorithm where the layers l can be interpreted as time steps of the iterative algorithm.

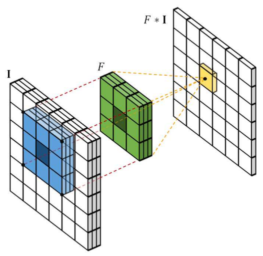


FIG. 12. Filtering or convolution: applying a filter of the size $3 \times 3 \times 3$ on an image of the size $6 \times 6 \times 3$ to get a filtered image or feature map of 6×6 (with proper boundary handling). Each pixel of the filtered image is computed by the weighted sum of the $3 \times 3 \times 3$ pixels of the input image centered at this pixel. There are 3 color channels (R, G, B), so both the input image and the filter are three-dimensional.

Convolution. The signal X can be an image, and the linear transformations at each layer may be convolutions with localized kernel functions (i.e. filters). That is, the row vectors of $W^{(l)}$ (as well as the elements of $b^{(l)}$) form different groups, and the vectors in the same group are localized and translation invariant versions of each other, like wavelets. Each group of vectors corresponds to a filter or a kernel or a channel. See Figures 12 and 13 for illustrations. Recent networks mostly use small filters of the size 3×3 [90, 92]. The minimal size 1×1 is also a popular choice [63, 92]. Such a filter fuses the features of different channels at the same location, and is often used for reducing or increasing the number of channels. When computing the filtered image, we can also sub-sample it by, e.g., taking one filter response every two pixels. The filter is said to have stride 2.

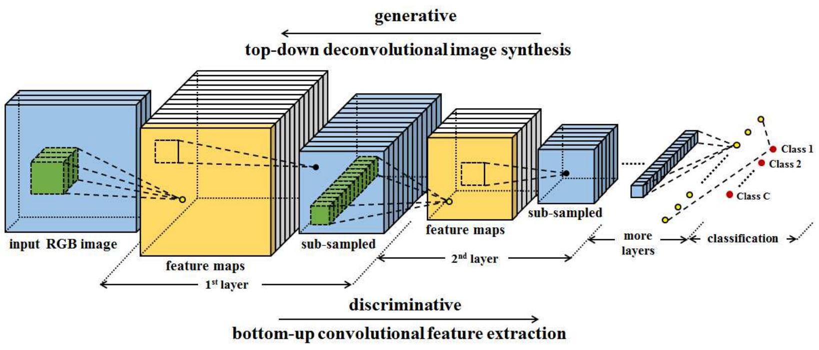


FIG. 13. Convolutional neural networks consist of multiple layers of filtering and sub-sampling operations for bottom-up feature extraction, resulting in multiple layers of feature maps and their sub-sampled versions. The top layer features are used for classification via multinomial logistic regression. The discriminative direction is from image to category, whereas the generative direction is from category to image.

4.3. *Descriptive models.* This subsection describes the hierarchical form of the descriptive models and the maximum likelihood learning algorithm.

We can generalize the descriptive model in the previous sections to a hierarchical form with multiple layers of features [72, 13, 106, 107],

$$X \rightarrow h^{(1)} \rightarrow ... \rightarrow h^{(L)} \rightarrow f_\theta(X) \tag{4.4}$$

which is a bottom-up process for computing $f_\theta(X)$, and θ collects all the weight and bias parameters at all the layers. The probability distribution is

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[f_\theta(X)\right] p_0(X), \tag{4.5}$$

where again $p_0(X)$ is the reference distribution such as Gaussian white noise model $p_0(X) \propto \exp\left(-\|X\|^2/2\sigma^2\right)$. Again the normalizing constant is $Z(\theta) = \int \exp(f_\theta(X)) p_0(X) dX = \mathrm{E}_{p_0}[\exp(f_\theta(X))]$. The energy function is

$$U_\theta(X) = \|X\|^2/2\sigma^2 - f_\theta(X). \tag{4.6}$$

$q_0(X)$ can also be a uniform distribution within a bounded range, then $U_\theta(X) = -f_\theta(X)$.

The model (4.5) can be considered a hierarchical generalization of the FRAME model. While the energy function of the FRAME model is defined in terms of element-wise non-linear functions of filter responses, model (4.5) involves recursions of this structure at multiple layers according to the ConvNet.

Suppose we observe training examples $\{X_i, i = 1, ..., n\}$. The maximum likelihood learning seeks to maximize $\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^n \log p_\theta(X_i)$. The gradient of $\mathcal{L}(\theta)$ is

$$\mathcal{L}'(\theta) \quad = \quad \frac{1}{n}\sum_{i=1}^n \frac{\partial}{\partial\theta} f_\theta(X_i) - \mathrm{E}_\theta\left[\frac{\partial}{\partial\theta} f_\theta(X)\right], \tag{4.7}$$

where E_θ denotes the expectation with respect to $p_\theta(X)$. The key identity underlying equation (4.7) is $d\log Z(\theta)/d\theta = \mathrm{E}_\theta[\partial f_\theta(X)/\partial\theta]$.

The expectation in equation (4.7) is analytically intractable and has to be approximated by MCMC, such as the Langevin dynamics, which samples from $p_\theta(X)$ by iterating the following step:

$$\begin{aligned} X_{\tau+1} &= X_\tau - \frac{s^2}{2}\frac{\partial}{\partial X}U_\theta(X_\tau) + s\mathcal{E}_\tau \quad (4.8)\\ &= X_\tau - \frac{s^2}{2}\left[\frac{X_\tau}{\sigma^2} - \frac{\partial}{\partial X}f_\theta(X_\tau)\right] + s\mathcal{E}_\tau, \quad (4.9)\end{aligned}$$

where τ indexes the time steps of the Langevin dynamics, s is the step size, and $\mathcal{E}_\tau \sim \mathrm{N}(0, I_p)$ is the Gaussian white noise term. A Metropolis-Hastings step can be added to correct for the finiteness of s. The Langevin dynamics was used by [115] for sampling from the linear form of the descriptive model such as the FRAME model.

We can run $\tilde{n}$ parallel chains of Langevin dynamics according to (4.9) to obtain the synthesized examples $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$. The Monte Carlo approximation to $\mathcal{L}'(\theta)$ is

$$\mathcal{L}'(\theta) \approx \frac{\partial}{\partial\theta}\left[\frac{1}{n}\sum_{i=1}^{n} f_\theta(X_i) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} f_\theta(\tilde{X}_i)\right], \quad (4.10)$$

which is the difference between the observed examples and the synthesized examples. We can then update $\theta^{(t+1)} = \theta^{(t)} + \eta_t\mathcal{L}'(\theta^{(t)})$, with $\mathcal{L}'(\theta^{(t)})$ computed according to (4.10). η_t is the learning rate. The convergence of this algorithm has been studied by [79, 108].

Alternating back-propagation: The learning and sampling algorithm is again an "analysis by synthesis" scheme. The sampling step runs the Langevin dynamics by computing $\partial f_\theta(X)/\partial X$, and the learning step updates θ by computing $\partial f_\theta(X)/\partial\theta$. Both derivatives can be computed by back-propagation, and they share the same computations of $\partial h^{(l)}/\partial h^{(l-1)}$.

Mode shifting interpretation: The data distribution $P_{\rm data}$ is likely to have many local modes. The $f_\theta(X)$ parametrized by the ConvNet can be flexible enough to creates many local modes to fit $P_{\rm data}$. We should learn $f_\theta(X)$ or equivalently the energy function $U_\theta(X)$ so that the energy function puts lower values on the observed examples than the unobserved examples. This is achieved by the learning and sampling algorithm, which can be interpreted as density shifting or mode shifting. In the sampling step, the Langevin dynamics settles the synthesized examples $\{\tilde{X}_i\}$ at the low energy regions or high density regions, or major modes (or basins) of $U_\theta(X)$, i.e., modes with low energies or high probabilities, so that $\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} U_\theta(\tilde{X}_i)$ tends to be low. The learning step seeks to change the energy function $U_\theta(X)$ by changing θ in order to increase $\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} U_\theta(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^{n} U_\theta(X_i)$. This has the effect of shifting the low energy or high density regions from the synthesized examples $\{\tilde{X}_i\}$ toward the observed examples $\{X_i\}$, or shifting the major modes of the energy function $U_\theta(X)$ from the synthesized examples toward the observed examples, until the observed examples reside in the major modes of the model. If the major modes are too diffused around the observed examples, the learning step will sharpen them to focus on the observed examples. This mode shifting interpretation is related to Hopfield network [45] and attractor network [86] with the Langevin dynamics serving as the attractor dynamics.

The energy landscape may have numerous major modes that are not occupied by the observed examples, and these modes imagine examples that are considered similar to the observed examples. While the maximum likelihood learning matches the average statistical properties between model and data, the ConvNet is expressive enough to create modes to encode the highly varied patterns. We still lack an in-depth understanding of the energy landscape.

Adversarial interpretation: The learning and sampling algorithm also has an adversarial interpretation where the learning and sampling steps play a minimax game. Let the value function be defined as

$$V = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} U_\theta(\tilde{X}_i) - \frac{1}{n} \sum_{i=1}^{n} U_\theta(X_i). \tag{4.11}$$

The learning step updates θ to increase V, while the Langevin sampling step tends to relax $\{\tilde{X}_i\}$ to decrease V. The zero temperature limit of the Langevin sampling is gradient descent that decreases V, and the resulting learning and sampling algorithm is a generalized version of herding [101]. See also [107]. This is related to Wasserstein GAN [3], but the critic and the actor are the same descriptive model, i.e., the model itself is its own generator and critic.

Multi-grid sampling and learning: In the high-dimensional space, e.g. image space, the model can be highly multi-modal. The MCMC in general and the Langevin dynamics in particular may have difficulty traversing different modes and it may be very time-consuming to converge. A simple and popular modification of the maximum likelihood learning is the contrastive divergence (CD) learning [40], where we obtain the synthesized example by initializing a finite-step MCMC from the observed example. The CD learning is related to score matching estimator [46, 47] and auto-encoder [99, 91, 1]. Such a method has the ability to handle large training datasets via mini-batch training. However, bias may be introduced in the learned model parameters in that the synthesized images can be far from the fair examples of the current model. A further modification of CD is persistent CD [95], where at the initial learning epoch the MCMC is still initialized from the observed examples, while in each subsequent learning epoch, the finite-step MCMC is initialized from the synthesized example of the previous epoch. The resulting synthesized examples can be less biased by the observed examples. However, the persistent chains may still have difficulty traversing different modes of the learned model.

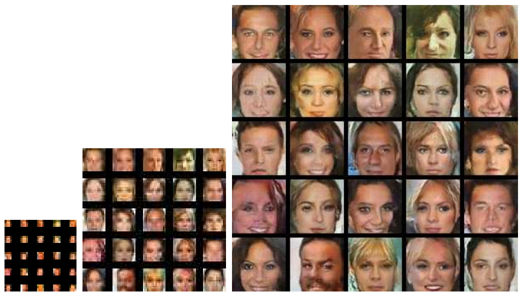


FIG. 14. Synthesized images at multi-grids [24]. From left to right: 4×4 grid, 16×16 grid and 64×64 grid. Synthesized image at each grid is obtained by 30 step Langevin sampling initialized from the synthesized image at the previous coarser grid, beginning with the 1×1 grid.

In [24], we developed a multi-grid sampling and learning method to address the above challenges under the constraint of finite budget MCMC. Specifically, we repeatedly down-scale each training image to get its multi-grid versions. Our method learns a separate descriptive model at each grid. Within each iteration of our learning algorithm, for each observed training image,

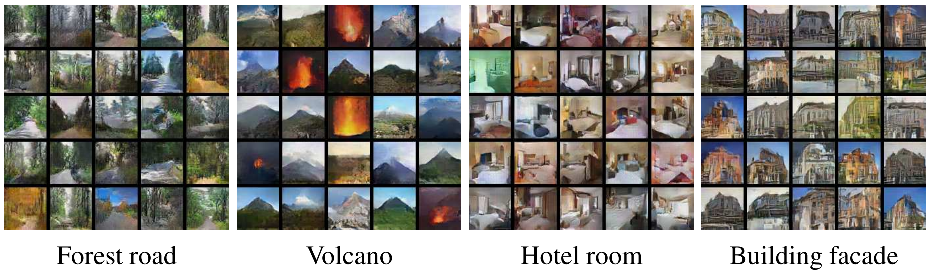


FIG. 15. Synthesized images from models learned by multi-grid method [24] from 4 categories of MIT places205 datasets.

we generate the corresponding synthesized images at multiple grids. Specifically, we initialize the finite-step MCMC sampling from the minimal 1×1 version of the training image, and the synthesized image at each grid serves to initialize the finite-step MCMC that samples from the model of the subsequent finer grid. See Figure 14 for an illustration, where we sample images sequentially at 3 grids, with 30 steps of Langevin dynamics at each grid. After obtaining the synthesized images at the multiple grids, the models at the multiple grids are updated separately and simultaneously based on the differences between the synthesized images and the observed training images at different grids.

Unlike original CD or persistent CD, the learned models are capable of generating new synthesized images from scratch with a fixed budget MCMC, because we only need to initialize the MCMC by sampling from the one-dimensional histogram of the 1×1 versions of the training images.

In our experiments, the training images are resized to 64×64. Since the models of the three grids act on images of different scales, we design a specific ConvNet structure per grid: grid1 has a 3-layer network with 5×5 stride 2 filters at the first layer and 3×3 stride 1 filters at the next two layers; grid2 has a 4-layer network with 5×5 stride 2 filters at the first layer and 3×3 stride 1 filters at the next three layers; grid3 has a 3-layer network with 5×5 stride 2 filters at the first layer, 3×3 stride 2 filters at the second layer, and 3×3 stride 1 filters at the third layer. Numbers of channels are $96-128-256$ at grid1 and grid3, and $96-128-256-512$ at grid2. A fully-connected layer with 1 channel output is added on top of every grid to get the value of the function $f_\theta(X)$. At each iteration, we run $l = 30$ steps of Langevin dynamics for each grid with step size $s = 0.3$. All networks are trained simultaneously with mini-batches of size 100 and an initial learning rate of 0.3. Learning rate is decayed logarithmically every 10 iterations.

We learn multi-grid models from several datasets including CelebA [66], MIT places205 [112] and CIFAR-10 [55]. In the CelebA dataset, we randomly sample 10,000 images for training. Figure 14 shows the synthesized examples. Figure 15 shows synthesized images from models learned from 4 categories of MIT places205 dataset by multi-grid method. We learn from each category separately. The number of training images is $15,100$ for each category.

Traditionally, the mixing time of Markov chain is defined via $d(t) = \max_x \|P^{(t)}(x,\cdot) - \pi\|_{\rm TV}$, where $P^{(t)}$ is the t-step transition, π is the stationary distribution, and $\|\cdot\|_{\rm TV}$ is the total variation distance. This is the worst case scenario by choosing the least favorable point mass at x. In our method, however, the initial distribution at each grid can be much more favorable, e.g., it may already agree approximately with π on the marginal distribution of the coarser grid, so that after t steps, the distribution of the sampled image can be close to π, even if this is not the case for the worst case starting point. Such non-persistent finite budget MCMC is computationally more manageable than persistent chains in learning.

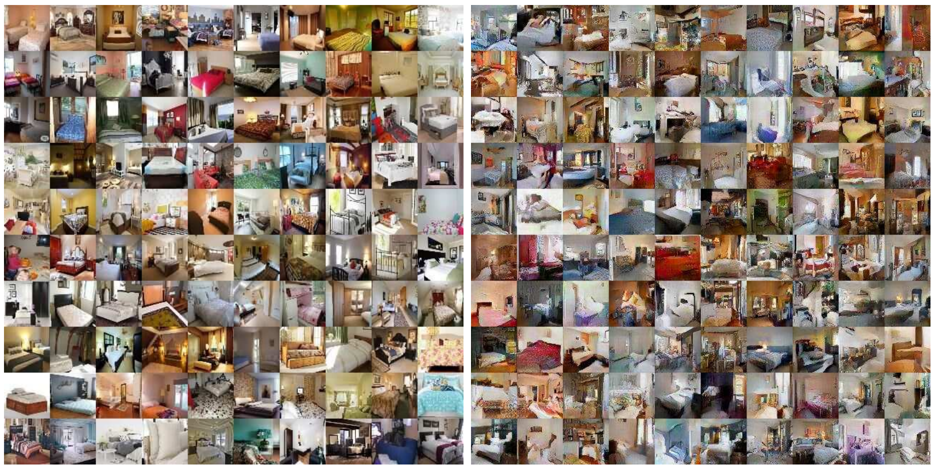


FIG. 16. Learning the multi-grid models from the LSUN bedroom dataset [24]. Left: random samples of training examples. Right: synthesized examples generated by the learned models.

To train multi-grid models on 10,000 training images for 400 iterations with a singe Titan X GPU, it takes about 7.45 hours. After training, it takes less than 1 second to generate a batch of 100 images. We also train the multi-grid models on LSUN bedroom dataset [109], which consists of roughly 3 million images. Figure 16 shows the learning results after 8 epochs.

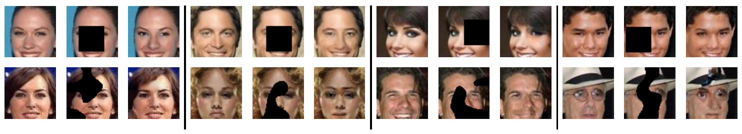


FIG. 17. Inpainting examples on CelebA dataset [24]. In each block from left to right: the original image; masked input; inpainted image by multi-grid method.

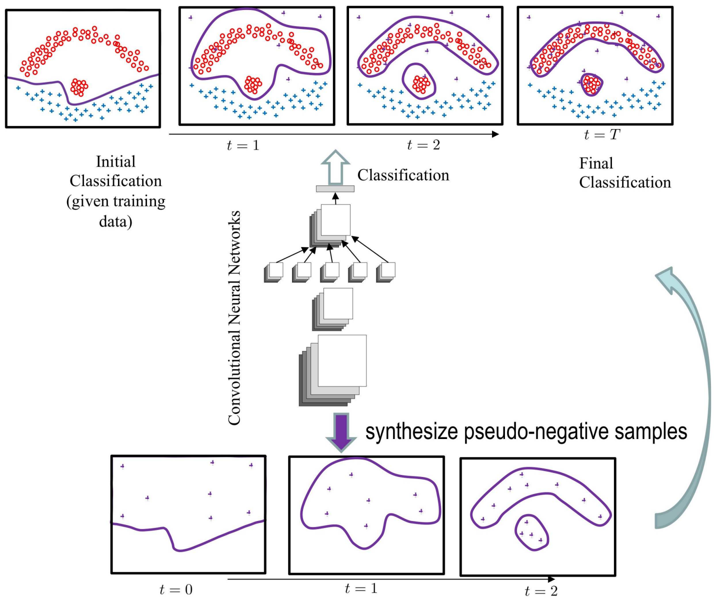
The learned descriptive model is a bottom-up ConvNet that consists of multiple layers of features. These features can be used for subsequent tasks such as classification. The learned models can also be used as a prior distribution for inpainting, as illustrated by Figure 17. See [24] for experiment details and numerical evaluations.

4.4. *Introspective learning.* This subsection describes the introspective learning method that learns the descriptive model by turning it into a discriminative model.

Model (4.5) corresponds to a classifier in the following sense [13, 106, 50]. Suppose there are K categories, $p_{\theta_k}(X)$, for $k = 1, ..., K$, in addition to the background category $p_0(X)$. The ConvNets $f_{\theta_k}(X)$ for $k = 1, ..., K$ may share common lower layers except the final layer for computing $f_{\theta_k}(X)$. Let ρ_k be the prior probability of category k, $k = 0, ..., K$. Then the posterior probability for classifying an example X to the category k is a soft-max multi-class classifier, i.e., the multinomial logistic regression:

$$\Pr(k|X) = \frac{\exp(f_{\theta_k}(X) + b_k)}{\sum_{k=0}^{K} \exp(f_{\theta_k}(X) + b_k)}, \tag{4.12}$$

where $b_k = \log(\rho_k/\rho_0) - \log Z(\theta_k)$, and for $k = 0$, $f_{\theta_0}(X) = 0$, $b_0 = 0$. Conversely, if we have the soft-max classifier (4.12), then the distribution of each category is $p_{\theta_k}(X)$ of the form (4.5). Thus the descriptive model directly corresponds to the commonly used discriminative ConvNet model.



$t = 1$
$t = 2$
$t = T$
Initial Classification (given training data)
Classification
Final Classification
Convolutional Neural Networks
synthesize pseudo-negative samples
$t = 0$
$t = 1$
$t = 2$

FIG. 18. Introspective learning [50]: The discriminative ConvNet trained on the observed examples versus the synthesized examples generated by the current model can be used to update the model and to generate new examples from the updated model.

In the case where we only observe unlabeled examples, we may model them by a single distribution $p_1(X) = p_\theta(X)$ in (4.5), and treat it as the positive distribution, and treat $p_0(X)$ as the negative distribution. Let ρ be the prior probability that a random example comes from p_1. Then the posterior probability that a random example X comes from p_1 is

$$\Pr(1|X) = \frac{1}{1 + \exp[-(f_\theta(X) + b)]}, \tag{4.13}$$

where $b = \log(\rho/(1-\rho)) - \log Z(\theta)$, i.e., a logistic regression.

Generalizing [96], [50] developed an introspective learning method for updating the model by learning a classifier or logistic regression to distinguish between the observed $\{X_i\}$ and the synthesized $\{\tilde{X}_i\}$, and tilt the current model according to the logistic regression. It is also an "analysis by synthesis" scheme as well as an adversarial scheme, except that the analysis is

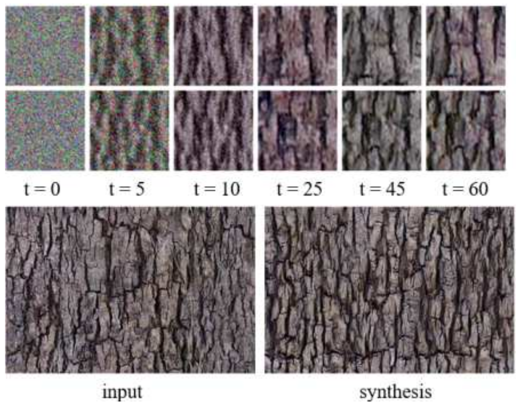


FIG. 19. Introspective learning [50]. Top row: patches of synthesized images in the introspective learning process. Bottom row: Left is the observed image. Right is the synthesized image generated by the learned model.

performed by a classifier. Specifically, let $p_t(X)$ be the current model. Each iteration of the introspective learning is as follows. The sampling step generates synthesized examples $\{\tilde{X}_i, i = 1, ..., n\}$ from $p_t(X)$. The learning step fits a logistic regression to separate the real examples $\{X_i, i = 1, ..., n\}$ from the synthesized examples $\{\tilde{X}_i, i = 1, ..., n\}$ to estimate $f_\theta(X)$ and b. Then we let $p_{t+1}(X) = \exp(f_\theta(X))p_t(X)/Z$, where $\log Z = -b$. [96, 50] show that p_t converges to P_{data} if the ConvNet is of infinite capacity. See Figure 18 for an illustration.

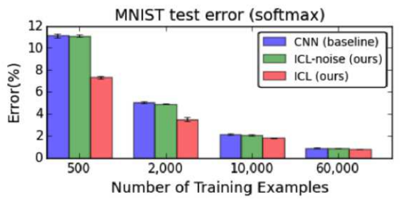


FIG. 20. Introspective learning [50] improves the classification performances, especially if the training dataset is of small or moderate size.

Numerical experiments in [50] show that the introspective method learns more accurate classifiers than purely discriminative methods in supervised learning, especially when the size of the training dataset is small or moderate. Figure 20 shows the results.

The introspective learning unifies the discriminative model and the descriptive model [50, 57, 62]. Unlike the generative adversarial networks (GAN) [29], the learned classifier is capable of introspection itself: it can be translated into a descriptive model to generate synthesized examples, without relying on a separate generative model.

4.5. *Generative models.* This subsection describes the hierarchical form of the generative models and the maximum likelihood learning algorithm.

We can generalize the generative model in the previous sections to a hierarchical form with multiple layers of hidden variables

$$h \to h^{(L)} \to ... \to h^{(1)} \to X, \tag{4.14}$$

which is a top-down process that transforms h to X. In the linear form of the generative model in the previous sections, the mapping from h to X is linear. In the hierarchical version, the mapping from h to X is a ConvNet defined by

$$h^{(l-1)} = g^{(l)}(W^{(l)}h^{(l)} + b^{(l)}), \tag{4.15}$$

for $l = L+1, ..., 1$, where $h^{(L+1)} = h$ and $h^{(0)} = X$. $g^{(l)}$ is the non-linear rectification function such as ReLU that is applied coordinate-wise. Let the resulting ConvNet be $X = g_\alpha(h)$, where $\alpha = (W^{(l)}, b^{(l)}, l = 1, 2, ..., L+1)$.

The top-down generative ConvNet was used by [110] to visualize the bottom-up ConvNet learned by the discriminative model. It was also used by [21] to learn a generative model of images of chairs, where the learning is supervised in that for each observed image of chair, a latent vector h is provided to specify the type of chair (represented by a one-hot vector) as well as view point and other geometric properties. The top-down ConvNet can learn accurate mapping from h to X, and the linear interpolation in the space of h leads to very realistic non-linear interpolation in the space of X.

The generative model can also be learned in unsupervised setting where h is unknown [29, 77, 53, 78, 68]. The model has the following form :

$$h \sim \mathrm{N}(0, I_d); \tag{4.16}$$
$$X = g_\alpha(h) + \epsilon; \epsilon \sim \mathrm{N}(0, \sigma^2 I_p), \tag{4.17}$$

where h is the d-dimensional hidden vector of latent factors, $g_\alpha(h)$ is a top-down ConvNet that maps the d-dimensional vector h to the p-dimensional signal X, where $d \leq p$. The model (4.17) is a generalization of factor analysis. While independent component analysis, sparse coding etc. generalize the prior distribution of factor analysis, the model (4.16) and (4.17) maintains the simple prior distribution of factor analysis, but generalizes the linear mapping in factor analysis to non-linear mapping parametrized by the top-down ConvNet (4.15). Like the word to vector representation [67], the hidden vector h may capture semantically meaningful information in the signal X.

The joint distribution

$$\begin{aligned} \log q_\alpha(h, X) &= \log\left[q(h)q_\alpha(X|h)\right] &(4.18)\\ &= -\frac{1}{2\sigma^2}\|X - g_\alpha(h)\|^2 - \frac{1}{2}\|h\|^2 + \text{const.} &(4.19)\end{aligned}$$

The marginal distribution $q_\alpha(X) = \int q_\alpha(h, X)dh$. The posterior distribution of the latent factors $q_\alpha(h|X) = q_\alpha(h, X)/q_\alpha(X) \propto q_\alpha(h, X)$. Here we use the notation q_α to denote the generative model in order to differentiate it from the descriptive model p_θ.

In our recent work [35], we study a maximum likelihood algorithm for learning the generative model (4.17) without resorting to an assisting network. Specifically, if we observe a training set of examples $\{X_i, i = 1, ..., n\}$, then each X_i has a corresponding latent h_i. We can train the generative model by maximizing the observed-data log-likelihood $\mathcal{L}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \log q_\alpha(X_i)$.

The gradient of $\mathcal{L}(\alpha)$ can be calculated according to the following identity:

$$\frac{\partial}{\partial \alpha} \log q_\alpha(X) = \frac{1}{q_\alpha(X)} \int \left[\frac{\partial}{\partial \alpha} \log q_\alpha(h, X) \right] q_\alpha(h, X) dh$$
$$= \mathrm{E}_{q_\alpha(h|X)} \left[\frac{\partial}{\partial \alpha} \log q_\alpha(X|h) \right]. \tag{4.20}$$

The expectation with respect to $q_\alpha(h|X)$ can be approximated by drawing samples from $q_\alpha(h|X)$ and then computing the Monte Carlo average.

The Langevin dynamics for sampling h from $p_\alpha(h|X)$ is

$$h_{\tau+1} = h_\tau + \frac{s^2}{2} \left[\frac{1}{\sigma^2}(X - g_\alpha(h_\tau)) \frac{\partial}{\partial h} g_\alpha(h_\tau) - h_\tau \right] + s\mathcal{E}_\tau, \tag{4.21}$$

where τ denotes the time step, s is the step size, and $\mathcal{E}_\tau \sim \mathrm{N}(0, I_d)$. Again we can add Metropolis-Hastings step to correct for the finiteness of s.

We can use stochastic gradient algorithm of [108] for learning, where in each iteration, for each X_i, h_i is sampled from $q_\alpha(h_i|X_i)$ by running a finite number of steps of Langevin dynamics starting from the current value of h_i. With the sampled $\{h_i\}$, we can update the parameters α based on the gradient $\mathcal{L}'(\alpha)$, whose Monte Carlo approximation is:

$$\mathcal{L}'(\alpha) \approx \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial \alpha} \log q_\alpha(X_i|h_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sigma^2}(X_i - g_\alpha(h_i)) \frac{\partial}{\partial \alpha} g_\alpha(h_i). \tag{4.22}$$

It is a non-linear regression of X_i on h_i. We update $\alpha^{(t+1)} = \alpha^{(t)} + \eta_t \mathcal{L}'(\alpha^{(t)})$, with $\mathcal{L}'(\alpha^{(t)})$ computed according to (4.22). η_t is the learning rate. The convergence of this algorithm follows [108].

Alternating back-propagation: Like the descriptive model, the maximum likelihood learning of the generative model (4.17) also follows the alternative back-propagation scheme. The Langevin dynamics for inference needs to compute $\partial g_\alpha(h)/\partial h$. The learning step needs to compute $\partial g_\alpha(h)/\partial \alpha$. Both gradients can be computed by back-propagation and they share the computations of $\partial h^{(l-1)}/\partial h^{(l)}$.

Our experiments show that the generative model is quite expressive. We adopt the structure of the generator network of [77, 21], where the top-down ConvNet consists of 5 layers.

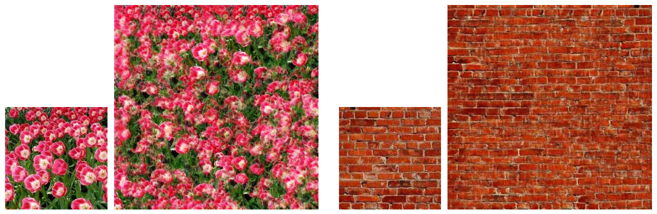


FIG. 21. Modeling texture patterns [35]. For each example, Left: the 224 × 224 observed image. Right: the 448 × 448 generated image.

Figure 21 shows the results of learning from texture images. We learn a separate model from each texture image. The images are collected from the Internet, and then resized to 224× 224. The synthesized images are 448 × 448.

The factors h at the top layer form a $\sqrt{d} \times \sqrt{d}$ image, with each pixel following $\mathrm{N}(0, 1)$ independently. The $\sqrt{d} \times \sqrt{d}$ image h is then transformed to X by the top-down ConvNet. We use $d = 7^2$ in the learning stage for all the texture experiments. In order to obtain the synthesized image, we randomly sample a 14 × 14 h from $\mathrm{N}(0, I)$, and then expand the learned network to generate the 448 × 448 synthesized image $g_\alpha(h)$.

The training network is as follows. Starting from 7 × 7 image h, the network has 5 layers of convolution with 5 × 5 kernels, with an up-sampling factor of 2 at each layer. The number of channels in the first layer is 512, and is decreased by a factor 2 at each layer. The Langevin steps $l = 10$ with step size $s = .1$.

experiment	$d = 20$	$d = 60$	$d = 100$	$d = 200$
Ours	.0810	.0617	.0549	.0523
PCA	.1038	.0820	.0722	.0621

TABLE 1. Reconstruction errors on testing images, using our method and PCA.

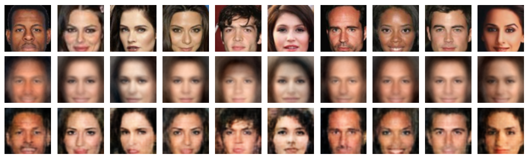


FIG. 22. Comparison between [35] and PCA. Row 1: original testing images. Row 2: reconstructions by PCA eigenvectors learned from training images. Row 3: Reconstructions by the generative model learned from training images. $d = 20$ for both methods.

The generative model performs non-linear dimension reduction which can be more flexible than linear dimension reduction such as principal component analysis (PCA) or factor analysis. After learning the model from the training images, we can evaluate how well the learned model can generalize by computing the reconstruction errors on the testing images. We randomly select 1000 face images for training and 300 images for testing from CelebA dataset. After learning, we infer the latent factors h for each testing image using Langevin dynamics, and then reconstruct the testing image by $g_\alpha(h)$ using the inferred h and the learned α. Table 1 shows the reconstruction error (measured by average per pixel difference relative to the range of the pixel intensities) of our method as compared to PCA learning for different latent dimensions d. Figure 22 shows some reconstructed testing images. For PCA, we learn the d eigenvectors from the training images, and then project the testing images on the learned eigenvectors for reconstruction.

In our recent work [38], we generalize the generative model for faces from multiple poses or views and learn the model from the Multi-PIE database [32]. Let $X_i^{(j)}$ be the j-th view of the i-th subject, for $j = 1, ..., m$. We can model $X_i^{(j)} = g_{\alpha_j}(h_i)$, where different views share the same latent vector h_i, but they are generated by different g_{α_j}. We can learn $(\alpha_j, j = 1, ..., m)$ using our learning algorithm. This enables us to change from one view to other views as illustrated by Figure 23.

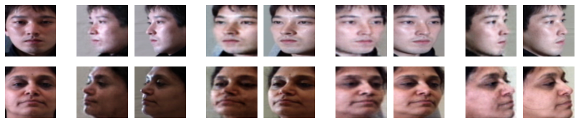


FIG. 23. Face rotation results on testing images [38]. First column: face image under standard pose (0°). Second to fifth column: each pair shows the rotated face by our method (left) and the ground truth target (right).

5. Variational, adversarial and cooperative learning. Both the descriptive model and the generative model involve intractable integrals. In the descriptive model, the normalizing constant is intractable. In the generative model, the marginal distribution of the observed signal is intractable. Consequently, the maximum likelihood learning algorithms of both models require MCMC sampling such as Langevin dynamics. To learn the descriptive model, we need to sample the synthesized examples. To learn the generative model, we need to sample the latent variables. It is possible to avoid MCMC sampling by variational and adversarial learning. It is also possible to speed up MCMC sampling by cooperative learning.

5.1. *Variational auto-encoder.* This subsection describes the variational learning of the generative model, where an inference model is learned to replace the MCMC sampling of the latent variables.

The maximum likelihood learning of the generative model seeks to minimize the divergence $\mathrm{KL}(P_{\mathrm{data}}(X)\|q_\alpha(X))$, where $q_\alpha(X) = \int q(h)q_\alpha(X|h)dh$ is the marginal distribution that is intractable. The variational auto-encoder (VAE) [53, 78, 68] changes the objective to

$$\min_\alpha \min_\phi \mathrm{KL}(P_{\mathrm{data}}(X)\rho_\phi(h|X)\|q(h)q_\alpha(X|h)), \tag{5.1}$$

where $\rho_\phi(h|X)$ is an analytically tractable approximation to $q_\alpha(h|X)$, and is called the inference model with parameter ϕ. Compared to the maximum likelihood objective $\mathrm{KL}(P_{\mathrm{data}}(X)\|q_\alpha(X))$, which is the KL-divergence between the marginal distributions of X, the VAE objective is the KL-divergence between the joint distributions of (h, X), i.e., $P_{\mathrm{data}}(X)\rho_\phi(h|X)$ and $q_\alpha(h, X) = q(h)q_\alpha(X|h)$, which is tractable because it does not involve the marginal $q_\alpha(X)$. The VAE objective is an upper bound of the maximum likelihood objective

$$\mathrm{KL}(P_{\mathrm{data}}(X)\rho_\phi(h|X)\|q_\alpha(h, X)) = \mathrm{KL}(P_{\mathrm{data}}(X)\|q_\alpha(X)) + \mathrm{KL}(\rho_\phi(h|X)\|q_\alpha(h|X)). \tag{5.2}$$

The accuracy of the VAE objective as an approximation to the maximum likelihood objective depends on the accuracy of the inference model $\rho_\phi(h|X)$ as an approximation to the true posterior distribution $q_\alpha(h|X)$.

For simplicity and slightly abusing the notation, write $P_{\mathrm{data}}(h, X) = P_{\mathrm{data}}(X)\rho_\phi(h|X)$, where P_{data} here is understood as the distribution of the complete data (h, X), with h imputed by $\rho_\phi(h|X)$, and $Q(h, X) = q(h)q_\alpha(X|h)$. The VAE is

$$\min_\alpha \min_\phi \mathrm{KL}(P_{\mathrm{data}}\|Q). \tag{5.3}$$

We can think of VAE from the perspective of alternating projection. (1) Fix α, find ϕ by minimizing $\mathrm{KL}(P_{\mathrm{data}}\|Q)$. This is to project the current Q onto the family of P_{data}. (2) Fix ϕ, find α by minimizing $\mathrm{KL}(P_{\mathrm{data}}\|Q)$. This is to project the current P_{data} onto the family of Q. Compared to the EM algorithm, projection (1) corresponds to the E-step to impute the missing data in the form of $\rho_\phi(h|X)$, and projection (2) corresponds to the M-step to fit the complete model $q(h)q_\alpha(X|h)$. The basic idea is illustrated by Figure 24.

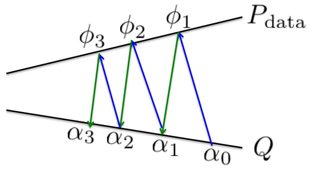


FIG. 24. VAE as alternating projection, where the straight lines illustrate the families of P_{data} and Q respectively, and each point is a distribution parametrized by ϕ or α.

The problem (5.1) is equivalent to maximizing

$$\mathrm{E}_{P_{\text{data}}}\left[\mathrm{E}_\phi[\log q_\alpha(h,X)] + \text{entropy}(\rho_\phi(h|X))\right] \tag{5.4}$$

$$= \mathrm{E}_{P_{\text{data}}}\left[\mathrm{E}_\phi[\log q_\alpha(X|h)] - \mathrm{KL}(\rho_\phi(h|X)\|q(h))\right] \tag{5.5}$$

where E_ϕ denotes the expectation with respect to $\rho_\phi(h|X)$, and $\mathrm{E}_{P_{\text{data}}}$ can be computed by averaging over the training examples. In (5.4) and (5.5), we have $q_\alpha(h,X)$ and $q_\alpha(X|h)$, as a result of merging $q_\alpha(X)$ and $q_\alpha(h|X)$ in (5.2), and both $q_\alpha(h,X)$ and $q_\alpha(X|h)$ are computationally tractable. If $\rho_\phi(h|X) = q_\alpha(h|X)$, then maximizing (5.4) with respect to α becomes the EM algorithm.

One popular choice of $\rho_\phi(h|X)$ is $\mathrm{N}(\mu_\phi(X), \sigma^2_\phi(X))$, where both $\mu_\phi(X)$ and $\sigma^2_\phi(X)$ can be represented by bottom-up neural networks with parameter ϕ.

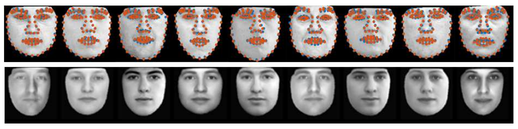


FIG. 25. Top Row: training images with landmarks. Bottom Row: synthesized images generated by the learned AAM model [37].

In our recent work [37], we show that VAE can replicate the active appearance model (AAM) [11]. Our experiments were inspired by a recent paper [7] that studied neurons in the middle lateral (ML) / middle fundus (MF) and anterior medial (AM) areas of the primate brain that are responsible for face recognition. Specifically, [7] recorded how these neurons respond to face stimuli generated by a pre-trained AAM model. We show that the observed properties of neurons' responses can be qualitatively replicated by VAE. The AAM model has an explicit shape representation in the form of landmarks, where the landmarks follow a shape model learned by principal component analysis. The faces can be aligned based on the landmarks, and the aligned faces follow an appearance model learned by another principal component analysis. The learning of the shape and appearance models require the landmarks in the training data. Figure 25 shows examples of face images to train AAM, and the synthesized face images from the trained AAM.

After learning the AAM model, we generate $20,000$ face images from the learned model. We then learn a VAE model from these images without the landmarks. Figure 26 displays test

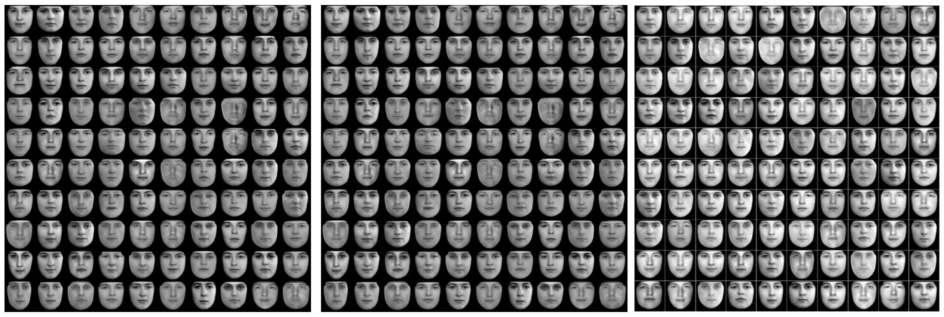


FIG. 26. Replicating AAM by VAE [37]. Left: test faces generated by AAM. Middle: reconstructed faces by the learned generative model. Right: synthesized images generated by the learned generative model.

images generated by the AAM model, their corresponding reconstructions by the learned VAE model, and the synthesized images generated by the learned VAE model.

In [7] the authors show that the responses from face patches ML/MF and AM have strong linear relationship with the shape and appearance variables in the original AAM model, where the responses of some neurons are highly correlated with the shape variables while the responses of other neurons are highly correlated with the appearance variables. In fact, one can further recover the original face images from the responses of these neurons, by linearly transforming the neurons' responses to the shape and appearance variables of the AAM, and then generating the image by the AAM variables. Apparently the neurons' responses form a code of the input face image that captures both the shape and appearance information of the input image. We show that the code learned by VAE, i.e., $\mu_\phi(X)$, has very strong linear relationship with the shape and appearance variables in AAM that generates X. The R^2 measure is over 96%. The biological observations found by [7] can be qualitatively reproduced by VAE. Even though the AAM model is highly non-linear due to shape deformation, the generative model has no difficulty replicating the AAM model without the supervision in the form of landmarks on the faces.

5.2. *Adversarial contrastive divergence.* This subsection describes the adversarial learning of the descriptive model, where a generative model is learned to replace the MCMC sampling of the descriptive model.

The maximum likelihood learning of the descriptive model seeks to minimize the divergence $\mathrm{KL}(P_{\mathrm{data}}(X)\|p_\theta(X))$, where the normalizing constant $Z(\theta)$ in p_θ is intractable. Recently [52] and [14] proposed to train the descriptive model p_θ and the generative model q_α jointly, which amounts to modifying the objective to

$$\min_\theta \max_\alpha [\mathrm{KL}(P_{\mathrm{data}}(X)\|p_\theta(X)) - \mathrm{KL}(q_\alpha(X)\|p_\theta(X))]. \tag{5.6}$$

See Figure 27 for an illustration. By maximizing over α, we minimize $\mathrm{KL}(q_\alpha(X)\|p_\theta(X))$, so that the objective function in (5.6) is a good approximation to $\mathrm{KL}(P_{\mathrm{data}}\|p_\theta)$. Because of the minimax nature of the objective, the learning is adversarial, where θ and α play a minimax game. While the generative model seeks to get close to the descriptive model, the descriptive model seeks to get close to the data distribution and to get away from the generative model. That is, the descriptive model can be considered a critic of the generative model by comparing it to the data distribution.

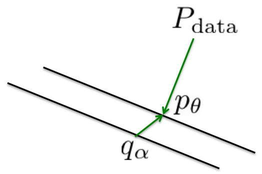


FIG. 27. Adversarial contrastive divergence. The straight lines illustrate the families of the descriptive and generative models, and each point is a probability distribution. The generative model seeks to approximate the descriptive model, while the descriptive model seeks to get close to the data distribution in contrast to the generative model.

The objective (5.6) contrasts interestingly with the objective for variational learning in (5.2). In the variational objective, we upper bound $\text{KL}(P_{\text{data}} \| q_\alpha)$ by adding another KL-divergence, so that we minimize over both α and ϕ. However, in the adversarial objective (5.6), we lower bound $\text{KL}(P_{\text{data}} \| p_\theta)$ by subtracting from it another KL-divergence, hence we need to find its saddle point. Thus the sign in front of the second KL-divergence determines whether it is variational learning or adversarial learning.

The adversarial objective (5.6) is also a form of contrastive divergence, except that the synthesized examples are provided by the generative model q_α directly, instead of being obtained by running a finite-step MCMC from the observed examples. We may call (5.6) the adversarial contrastive divergence. It is equivalent to

$$\min_\theta \max_\alpha \left[\text{E}_{P_{\text{data}}}[U_\theta(X)] - \text{E}_\theta[U_\theta(X)] + \text{entropy}(q_\alpha) \right], \tag{5.7}$$

which is the form proposed by [14]. In this form, the $\log Z(\theta)$ term is canceled out, so that we do not have to deal with this intractable term.

However, the entropy term $\text{entropy}(q_\alpha)$ or the second KL-divergence in (5.6) is not in closed form, and still needs approximation. We can again use the variational approach to approximate $\text{KL}(q_\alpha(X) \| p_\theta(X))$ by

$$\text{KL}(q_\alpha(X) \| p_\theta(X)) + \text{KL}(q_\alpha(h|X) \| \rho_\phi(h|X)) = \text{KL}(q_\alpha(h, X) \| p_\theta(X)\rho_\phi(h|X)), \tag{5.8}$$

where $\rho_\phi(h|X)$ is again a learned inference model. This leads to the method used by [14]. Again we only need to deal with the tractable joint model $q_\alpha(h, X)$. Thus the learning problem becomes

$$\min_\theta \max_\alpha \max_\phi [\text{KL}(P_{\text{data}}(X) \| p_\theta(X)) - \text{KL}(q_\alpha(h, X) \| p_\theta(X)\rho_\phi(h|X))]. \tag{5.9}$$

There are three networks that need to be learned, including the descriptive model p_θ, the generative model q_α, and the inference model ρ_ϕ. Write $P_{\text{data}}(h, X) = P_{\text{data}}(X)\rho_\phi(h|X)$, $Q(h, X) = q(h)q_\alpha(X|h)$, and $P(h, X) = p_\theta(X)\rho_\phi(h|X)$. The above objective is

$$\min_\theta \max_\alpha \max_\phi [\text{KL}(P_{\text{data}} \| P) - \text{KL}(Q \| P)]. \tag{5.10}$$

Compared to the variational learning in (5.2), $\rho_\phi(h|X)$ appears on the left side of KL-divergence in (5.2), but it appears on the right side of KL-divergence in (5.9). The learning of $\rho_\phi(h|X)$ is from the synthesized data generated by $q_\alpha(h, X)$ instead of real data. This is similar to the sleep phase of the wake-sleep algorithm [41].

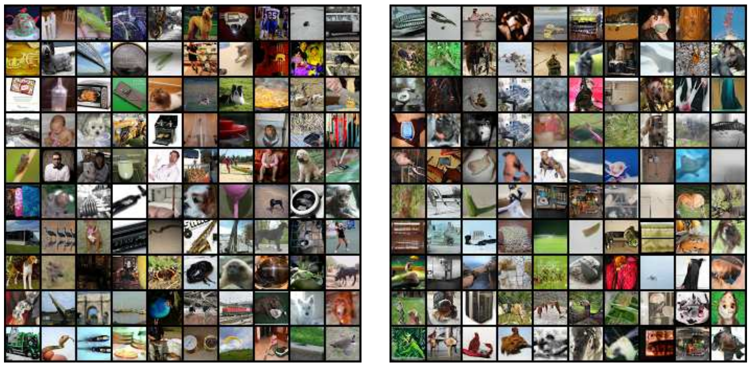


FIG. 28. Learning the models from the ImageNet dataset. Left: random samples of training examples. Right: synthesized examples generated by the learned generative model.

We train the three nets on the down-sampled 32x32 imageNet dataset [18] (roughly 1 million images). For the generative model, starting from the latent vector h of 100 dimensions, we use 5 layers of kernels of stride 2, where the sizes of kernels of the first 4 layers are 4×4, and the size of the kernels of the bottom layer is 3×3. The numbers of channels at these layers are 512, 512, 256, 128, 3 respectively. Each layer is followed by batch normalization and ReLU non-linearity, except the last layer where tanh is used. For the inference model, we use the mirror structure as the generative model. We build the last layer separately to model the posterior mean and variance. For the descriptive model, we use the same structure as the inference net.

Figure 28 displays the learning results, where the left panel shows randomly selected training examples and the right panel shows the random examples generated by the learned generative model.

Another possibility of adversarial contrastive divergence learning is to learn a joint energy-based model $p_\theta(h, X)$ by

$$\min_\theta \min_\phi \max_\alpha [\mathrm{KL}(P_{\text{data}}(X)\rho_\phi(h|X) \| p_\theta(h, X)) - \mathrm{KL}(q(h)q_\alpha(X|h) \| p_\theta(h, X))]. \quad (5.11)$$

5.3. *Integrating variational and adversarial learning.* We can integrate or unify the variational and adversarial learning methods.

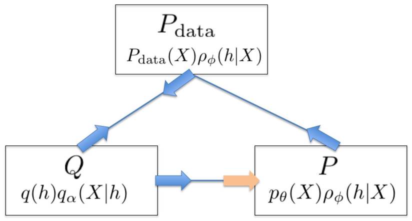

P_{data}
$P_{\text{data}}(X)\rho_\phi(h|X)$
Q
$q(h)q_\alpha(X|h)$
P
$p_\theta(X)\rho_\phi(h|X)$

FIG. 29. Divergence triangle [36]. The generative model seeks to get close to the data distribution as well as the descriptive model. The descriptive model seeks to get close to the data distribution and get away from the generative model.

Following the notation of previous subsections, write $P_{\text{data}}(h, X) = P_{\text{data}}(X)\rho_\phi(h|X)$, $P(h, X) = p_\theta(X)\rho_\phi(h|X)$, and $Q(h, X) = q(h)q_\alpha(X|h)$. It has been noticed by the recent work [36] that the variational objective $\text{KL}(P_{\text{data}}\|Q)$ and the adversarial objective $\text{KL}(P_{\text{data}}\|P) - \text{KL}(Q\|P)$ can be combined into

$$\max_\theta \min_\alpha \min_\phi [\text{KL}(P_{\text{data}}\|Q) + \text{KL}(Q\|P) - \text{KL}(P_{\text{data}}\|P)], \tag{5.12}$$

which is in the form of a triangle formed by P_{data}, P, and Q. See Figure 29 for an illustration. As shown by [36], one can learn the descriptive model, the generative model, and the inference model jointly using the above objective.

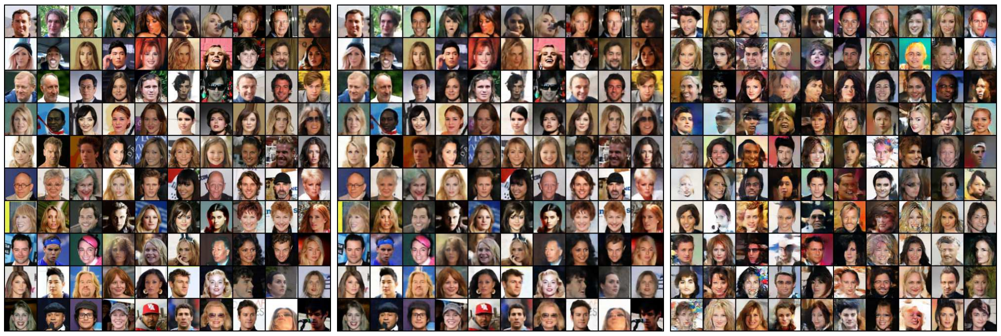


FIG. 30. Learning the models from CelebA dataset [36]. From left to right: original images, reconstructed images, and generated images.

Figure 30 displays an example in [36] where the models are learned from the CelebA dataset. The left panel shows some random training examples, the middle panel shows the corresponding reconstructed examples by the learned inference model, and the right panel shows some examples generated by the learned generative model.

5.4. *Cooperative learning.* This subsection describes the cooperative training of the descriptive and generative models which jumpstart each other's MCMC sampling.

We can learn the descriptive model and the generative model separately, and we have been able to scale up the learning to big datasets. However, the separate learning algorithms can still be slow due to MCMC sampling. Recently we discover that we can speed up the learning of the two models by coupling the two maximum likelihood learning algorithms into a cooperative algorithm that we call the CoopNets algorithm [104, 105]. It is based on the following two key observations. (1) The generative model can generate examples directly, so we can use it as an approximate sampler of the descriptive model. (2) The generative model can be learned more easily if the latent factors are known, which is the case with the synthesized examples.

Generative model as a sampler. The generative model can serve as an approximate sampler of the descriptive model. To sample from the descriptive model, we can initialize the synthesized examples by generating examples from the generative model. We first generate $\hat{h}_i \sim \text{N}(0, I_d)$, and then generate $\hat{X}_i = g(\hat{h}_i; \alpha) + \epsilon_i$, for $i = 1, ..., \tilde{n}$. If the current generative model q_α is close to the current descriptive model p_θ, then the generated $\{\hat{X}_i\}$ should be a good initialization for sampling from the descriptive model, i.e., starting from the $\{\hat{X}_i, i = 1, ..., \tilde{n}\}$, we run Langevin dynamics for l steps to get $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$, which are revised versions of $\{\hat{X}_i\}$. These $\{\tilde{X}_i\}$

can be used as the synthesized examples from the descriptive model. We can then update θ in the same way as we learn the descriptive model.

MCMC teaching. The descriptive model can teach the generative model via MCMC. In order to update α of the generative model, we treat the $\{\tilde{X}_i, i = 1, ..., \tilde{n}\}$ produced by the above procedure as the training data for the generative model. Since these $\{\tilde{X}_i\}$ are obtained by the Langevin dynamics initialized from $\{\hat{X}_i, i = 1, ..., \tilde{n}\}$, which are generated by the generative model with known latent factors $\{\hat{h}_i, i = 1, ..., \tilde{n}\}$, we can update α by learning from $\{(\hat{h}_i, \tilde{X}_i), i = 1, ..., \tilde{n}\}$, which is a supervised learning problem, or more specifically, a non-linear regression of $\tilde{X}_i$ on $\hat{h}_i$. At $\alpha^{(t)}$, the latent vector $\hat{h}_i$ generates and thus reconstructs the initial example $\hat{X}_i$. After updating α, we want $\hat{h}_i$ to reconstruct the revised example $\tilde{X}_i$. That is, we revise α to absorb the MCMC transition from $\hat{X}_i$ to $\tilde{X}_i$ for sampling the descriptive model, so that the generative model shifts its density from $\{\hat{X}_i\}$ to $\{\tilde{X}_i\}$. The left diagram in (5.13) illustrates the basic idea.

$$\begin{tikzcd}
\hat{h}_i \arrow[d, Rightarrow, "\alpha^{(t)}"'] \arrow[dr, Rightarrow, "\alpha^{(t+1)}"] & \\
\hat{X}_i \arrow[r, dashed, "\theta^{(t)}"'] & \tilde{X}_i
\end{tikzcd}
\qquad
\begin{tikzcd}
\hat{h}_i \arrow[r, dashed, "\alpha^{(t)}"] \arrow[d, Rightarrow, "\alpha^{(t)}"'] & \tilde{h}_i \arrow[d, Rightarrow, "\alpha^{(t+1)}"] \\
\hat{X}_i \arrow[r, dashed, "\theta^{(t)}"'] & \tilde{X}_i
\end{tikzcd}
\tag{5.13}$$

In the two diagrams in (5.13), the double-line arrows indicate generation and reconstruction by the generative model, while the dashed-line arrows indicate Langevin dynamics for MCMC sampling and inference in the two models. The diagram on the right in (5.13) illustrates a more rigorous method, where we initialize the Langevin inference of $\{h_i, i = 1, ..., \tilde{n}\}$ from $\{\hat{h}_i\}$, and then update α based on $\{(\tilde{h}_i, \tilde{X}_i), i = 1, ..., \tilde{n}\}$. The diagram on the right shows how the two models jumpstart each other's MCMC.

The learning of the descriptive model is based on the modified contrastive divergence,

$$\mathrm{KL}(P_{\mathrm{data}} \| p_\theta) - \mathrm{KL}(M_\theta q_\alpha \| p_\theta), \tag{5.14}$$

where q_α provides the initialization of the finite-step MCMC, whose transition kernel is denoted M_θ, and $M_\theta q_\alpha$ denotes the marginal distribution obtained after running M_θ from q_α. The learning of the generative model is based on how $M_\theta q_\alpha$ modifies q_α, and is accomplished by $\min_{q_\alpha} \mathrm{KL}(M_\theta q_{\alpha^{(t)}} \| q_\alpha)$. In the idealized case of infinite capacity of q_α so that the KL-divergence can be minimized to zero, the learned q_α will satisfy $q_\alpha = M_\theta q_\alpha$, i.e., q_α is the stationary distribution of M_θ. But the stationary distribution of M_θ is nothing but p_θ. Thus the learned q_α will be the same as q_θ. Then the second KL-divergence in (5.14) will become zero, and the learning of the descriptive model is to minimize $\mathrm{KL}(P_{\mathrm{data}} \| p_\theta)$, which is maximum likelihood.

We conduct experiments on learning from Imagenet dataset [18]. We adopt a 4-layer descriptive model and a 5-layer generative model. We set the number of Langevin dynamics steps in each learning iteration to $l = 10$. The number of learning iterations is $1,000$. After learning the models, we synthesize images using the learned models.

In our first experiment, we learn from images that are randomly sampled from 10 Imagenet scene categories. The number of images sampled from each category is 1100. We mix the images

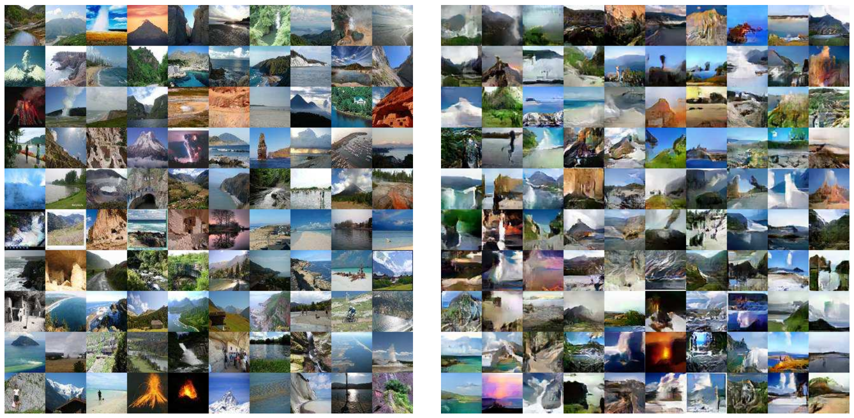


FIG. 31. Cooperative learning [104, 105]. The training set consists of 11,000 images (64 × 64) randomly sampled from 10 Imagenet scene categories. Left panel: random samples of training images. Right panel: random samples generated by the learned models.

from all the categories as a single training set. Figure 31 displays the observed examples randomly sampled from the training set, and the synthesized examples generated by the CoopNets algorithm.

Figure 32 shows 4 examples of interpolating between latent vectors h. For each row, the images at the two ends are generated by h vectors randomly sampled from $\mathrm{N}(0, I_d)$. Each image in the middle is obtained by first interpolating the h vectors of the two end images, and then generating the image using the learned models. This experiment shows that we learn smooth generative model that traces the manifold of the data distribution.

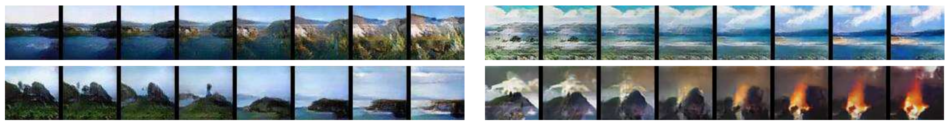


FIG. 32. Cooperative learning [104, 105]. Interpolation between latent vectors of the images on the two ends.

We evaluate the synthesis quality by the Inception score [85]. Our method is competitive to DCGAN [77], EBGAN [111], Wasserstein GAN [3], InfoGAN [9], VAE [53], the method of [52].

Compared to the three nets in [14], the cooperative learning method only needs two nets. Moreover, the finite-step MCMC serves to bridge the generative model and the descriptive model, so that the synthesized examples are closer to fair samples from the descriptive model.

6. Discussion. To summarize the relationships between the non-hierarchical linear forms and the hierarchical non-linear forms of the three families of models, the non-hierarchical form has one layer of features or hidden variables, and they are designed. The hierarchical form has multiple layers of features or hidden variables, and all the layers are learned from the data.

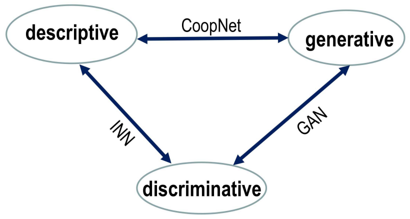


FIG. 33. The connections between the three families of models. The discriminative and the generative models are connected by the generative adversarial networks (GAN). The discriminative and the descriptive models are connected by the introspective neural networks (INN). The descriptive and the generative models are connected by cooperative learning.

To summarize the relationships between the three families of models, we have the following connections:

(1) The discriminative model and the descriptive model can be translated into each other by the Bayes rule. The introspective learning method unifies the two models.
(2) The descriptive model and the generative model can be learned together by adversarial contrastive divergence or the cooperative learning method via MCMC teaching.
(3) The discriminative model and the generative model can be learned together by adversarial training.

See Figure 33 for an illustration.

Besides the models reviewed in this paper, there are other probabilistic models, such as the deep Boltzmann machine [42, 84, 61], which is an energy-based model with multiple layers of latent variables, auto-regressive models [74], the deep generalizations of the independent component analysis model [19, 20].

In the cooperative learning, the descriptive model and the generative model are parametrized by separate networks. It is more desirable to integrate the two classes of models within a common network.

The existing models are still quite far from what Grenander might have searched for, in that they are still more or less black box models with ConvNet parametrizations. A more interpretable model is the And-Or graph [116], which alternates between layers of And nodes and Or nodes. An And node models the composition of parts, while an Or node models the alternative choices of parts according to a certain probability distribution. Such an And-Or grammar can generalize to unseen patterns by reconfiguration of parts. In fact the neural network can be interpreted as a dense version of And-Or graph in that the linear weighted sum can be interpreted as And nodes and the rectification and max pooling can be interpreted as Or nodes. Figure 34 shows an example of And-Or template of animal faces [88].

Ideally, as illustrated by Figure 35, we should have simple descriptive and generative models at the lowest layers, with the descriptive models accounting for high dimensional or high entropy patterns such as stochastic textures, and the generative models accounting for low-dimensional or low entropy patterns such as textons. In the middle layers we should have stochastic grammars to define the explicit compositional patterns of objects and their parts, as well as their relations [27, 116]. At the top layer, we should have logical reasoning based on the learned common sense

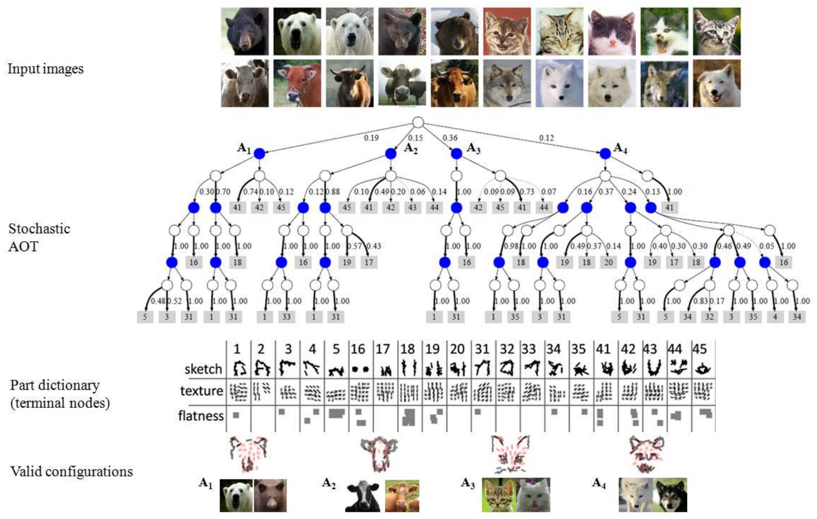


FIG. 34. And-Or template [88] for modeling recursive compositions of alternative parts: Each And node (blue circle) is a composition of some Or nodes. Each Or node (blank circle) is a probability distribution over some And nodes. An And node models the composition of parts. An Or node models the alternative choices of each part.

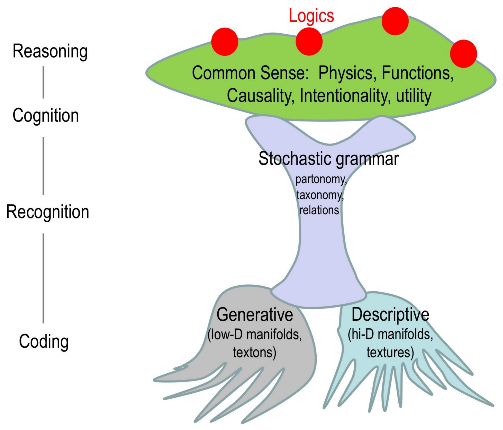

Logics
Reasoning
Cognition
Recognition
Coding
Common Sense: Physics, Functions, Causality, Intentionality, utility
Stochastic grammar
partonomy, taxonomy, relations
Generative (low-D manifolds, textons)
Descriptive (hi-D manifolds, textures)

FIG. 35. Hierarchical representation of patterns, with simple generative and descriptive models for textons and textures at the lower layers, the stochastic grammar in the middle layers, and logic reasoning with common sense at higher layers.

about physics, funtionality and causality. It is our hope that a unified model of this form can be developed in the future.

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

REFERENCES

[1] Guillaume Alain and Yoshua Bengio, *What regularized auto-encoders learn from the data-generating distribution*, The Journal of Machine Learning Research **15** (2014), no. 1, 3563–3593.

[2] Shun-ichi Amari and Hiroshi Nagaoka, *Methods of information geometry*, vol. 191, American Mathematical Soc., 2007.

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

[4] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville, *Deep learning*, Book in preparation for MIT Press, 2015.

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

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

[7] Le Chang and Doris Y Tsao, *The code for facial identity in the primate brain*, Cell **169** (2017), no. 6, 1013–1028.

[8] Scott Shaobing Chen, David L Donoho, and Michael A Saunders, *Atomic decomposition by basis pursuit*, SIAM Journal on Scientific Computing **20** (1998), no. 1, 33–61.

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

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

[11] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor, *Active appearance models*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2001), no. 6, 681–685.

[12] Corinna Cortes and Vladimir Vapnik, *Support-vector networks*, Machine learning **20** (1995), no. 3, 273–297.

[13] Jifeng Dai, Yang Lu, and Ying-Nian Wu, *Generative modeling of convolutional neural networks*, arXiv preprint arXiv:1412.6296 (2014).

[14] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville, *Calibrating energy-based generative adversarial networks*, arXiv preprint arXiv:1702.01691 (2017).

[15] John G Daugman, *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters*, JOSA A **2** (1985), no. 7, 1160–1169.

[16] Stephen Della Pietra, Vincent Della Pietra, and John Lafferty, *Inducing features of random fields*, IEEE Transactions on Pattern Analysis and Machine Intelligence (1997), no. 4, 380–393.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin, *Maximum likelihood from incomplete data via the em algorithm*, Journal of the royal statistical society. Series B (methodological) (1977), 1–38.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

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

[20] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio, *Density estimation using real nvp*, arXiv preprint arXiv:1605.08803 (2016).

[21] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox, *Learning to generate chairs with convolutional neural networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1538–1546.

[22] Yoav Freund and Robert E Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of computer and system sciences **55** (1997), no. 1, 119–139.

[23] Jerome H Friedman, *Multivariate adaptive regression splines*, The annals of statistics (1991), 1–67.

[24] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu, *Learning generative convnets via multigrid modeling and sampling*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9155–9164.

[25] Stuart Geman and Donald Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence (1997), no. 4, 380–393.

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

[27] Stuart Geman, Daniel F Potter, and Zhiyi Chi, *Composition systems*, Quarterly of Applied Mathematics **60** (2002), no. 4, 707–736.

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

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[30] Ulf Grenander, *A unified approach to pattern analysis*, Advances in Computers **10** (1970), 175–216.
[31] Ulf Grenander and Michael I Miller, *Pattern theory: from representation to inference*, Oxford University Press, 2007.
[32] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker, *Multi-pie*, Image Vision Comput. **28** (2010), no. 5, 807–813.
[33] Cheng-En Guo, Song-Chun Zhu, and Ying Nian Wu, *Modeling visual patterns by integrating descriptive and generative methods*, International Journal of Computer Vision **53** (2003), no. 1, 5–29.
[34] Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu, *Primal sketch: Integrating structure and texture*, Computer Vision and Image Understanding **106** (2007), no. 1, 5–19.
[35] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu, *Alternating back-propagation for generator network.*, AAAI, vol. 3, 2017, p. 13.
[36] Tian Han, Erik Nijkamp, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu, *Divergence triangle for joint training of energy-based model, generator model and inference model*, (2018).
[37] Tian Han, Jiawen Wu, and Ying Nian Wu, *Replicating active appearance model by generator network*, International Joint Conferences on Artificial Intelligence, 2018.
[38] Tian Han, Xianglei Xing, and Ying Nian Wu, *Learning multi-view generator network for shared representation*, International Conference on Pattern Recognition, 2018.
[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[40] Geoffrey E Hinton, *Training products of experts by minimizing contrastive divergence.*, Neural Computation **14** (2002), no. 8, 1771–1800.
[41] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal, *The "wake-sleep" algorithm for unsupervised neural networks*, Science **268** (1995), no. 5214, 1158–1161.
[42] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, *A fast learning algorithm for deep belief nets*, Neural Computation **18** (2006), no. 7, 1527–1554.
[43] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), no. 8, 1735–1780.
[44] Yi Hong, Zhangzhang Si, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu, *Unsupervised learning of compositional sparse code for natural image representation*, Quarterly of Applied Mathematics **72** (2013), 373–406.
[45] John J Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the national academy of sciences **79** (1982), no. 8, 2554–2558.
[46] Aapo Hyvärinen, *Estimation of non-normalized statistical models by score matching*, Journal of Machine Learning Research **6** (2005), 695–709.
[47] Aapo Hyvarinen, *Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables*, IEEE Transactions on neural networks **18** (2007), no. 5, 1529–1531.
[48] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja, *Independent component analysis*, vol. 46, John Wiley & Sons, 2004.
[49] Sergey Ioffe and Christian Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv preprint arXiv:1502.03167 (2015).
[50] Long Jin, Justin Lazarow, and Zhuowen Tu, *Introspective learning for discriminative classification*, Advances in Neural Information Processing Systems, 2017.
[51] Bela Julesz et al., *Textons, the elements of texture perception, and their interactions*, Nature **290** (1981), no. 5802, 91–97.
[52] Taesup Kim and Yoshua Bengio, *Deep directed generative models with energy-based probability estimation*, arXiv preprint arXiv:1606.03439 (2016).
[53] Diederik P. Kingma and Max Welling, *Auto-encoding variational bayes*, International Conference for Learning Representations (2014).
[54] Yehuda Koren, Robert Bell, and Chris Volinsky, *Matrix factorization techniques for recommender systems*, Computer (2009), no. 8, 30–37.
[55] Alex Krizhevsky and Geoffrey E Hinton, *Learning multiple layers of features from tiny images*, (2009).
[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[57] Justin Lazarow, Long Jin, and Zhuowen Tu, *Introspective neural networks for generative modeling*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2774–2783.

[58] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

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

[60] Daniel D Lee and H Sebastian Seung, *Algorithms for non-negative matrix factorization*, Advances in Neural Information Processing Systems, 2001, pp. 556–562.

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

[62] Kwonjoon Lee, Weijian Xu, Fan Fan, and Zhuowen Tu, *Wasserstein introspective neural networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[63] Min Lin, Qiang Chen, and Shuicheng Yan, *Network in network*, arXiv preprint arXiv:1312.4400 (2013).

[64] Ce Liu, Song-Chun Zhu, and Heung-Yeung Shum, *Learning inhomogeneous gibbs model of faces by minimax entropy*, International Conference on Computer Vision, 2001, pp. 281–287.

[65] Jun S Liu, *Monte carlo strategies in scientific computing*, Springer Science & Business Media, 2008.

[66] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, *Deep learning face attributes in the wild*, International Conference on Computer Vision, 2015, pp. 3730–3738.

[67] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, *Distributed representations of words and phrases and their compositionality*, Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.

[68] Andriy Mnih and Karol Gregor, *Neural variational inference and learning in belief networks*, International Conference on Machine Learning, 2014, pp. 1791–1799.

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

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

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

[72] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng, *Learning deep energy models*, International Conference on Machine Learning, 2011, pp. 1105–1112.

[73] Bruno A Olshausen and David J Field, *Sparse coding with an overcomplete basis set: A strategy employed by v1?*, Vision Research **37** (1997), no. 23, 3311–3325.

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

[75] Pentti Paatero and Unto Tapper, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics **5** (1994), no. 2, 111–126.

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

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

[78] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra, *Stochastic backpropagation and approximate inference in deep generative models*, International Conference on Machine Learning, 2014, pp. 1278–1286.

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

[80] Stefan Roth and Michael J Black, *Fields of experts: A framework for learning image priors*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 860–867.

[81] Sam T Roweis and Lawrence K Saul, *Nonlinear dimensionality reduction by locally linear embedding*, Science **290** (2000), no. 5500, 2323–2326.

[82] Donald B Rubin, *Multiple imputation for nonresponse in surveys*, vol. 81, John Wiley & Sons, 2004.

[83] Donald B Rubin and Dorothy T Thayer, *Em algorithms for ml factor analysis*, Psychometrika **47** (1982), no. 1, 69–76.

[84] Ruslan Salakhutdinov and Geoffrey E Hinton, *Deep boltzmann machines*, International Conference on Artificial Intelligence and Statistics, 2009.

[85] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, *Improved techniques for training gans*, Advances in Neural Information Processing Systems, 2016, pp. 2226–2234.

[86] H. Sebastian Seung, *Learning continuous attractors in recurrent networks*, Advances in Neural Information Processing Systems, 1998, pp. 654–660.

[87] Zhangzhang Si and Song-Chun Zhu, *Learning hybrid image templates (hit) by information projection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **99** (2011), no. 7, 1354–1367.
[88] Zhangzhang Si and Song-Chun Zhu, *Learning and-or templates for object recognition and detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), no. 9, 2189–2205.
[89] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al., *Mastering the game of go without human knowledge*, Nature **550** (2017), no. 7676, 354–359.
[90] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, ICLR (2015).
[91] Kevin Swersky, Marc'Aurelio Ranzato, David Buchman, Benjamin Marlin, and Nando Freitas, *On autoencoders and score matching for energy based models*, ICML, ACM, 2011, pp. 1201–1208.
[92] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, *Rethinking the inception architecture for computer vision*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
[93] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton, *Energy-based models for sparse overcomplete representations*, Journal of Machine Learning Research **4** (2003), no. Dec, 1235–1260.
[94] Robert Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodological) **58** (1996), no. 1, 267–288.
[95] Tijmen Tieleman, *Training restricted boltzmann machines using approximations to the likelihood gradient*, International Conference on Machine Learning, 2008, pp. 1064–1071.
[96] Zhuowen Tu, *Learning generative models via discriminative approaches*, 2007 IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
[97] Zhuowen Tu and Song-Chun Zhu, *Image segmentation by data-driven markov chain monte carlo*, IEEE Transactions on pattern analysis and machine intelligence **24** (2002), no. 5, 657–673.
[98] ———, *Parsing images into regions, curves and curve groups*, International Journal of Computer Vision **69** (2006), no. 2, 223–249.
[99] Pascal Vincent, *A connection between score matching and denoising autoencoders*, Neural Computation **23** (2011), no. 7, 1661–1674.
[100] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, *Extracting and composing robust features with denoising autoencoders*, International Conference on Machine Learning, 2008, pp. 1096–1103.
[101] Max Welling, *Herding dynamical weights to learn*, International Conference on Machine Learning, 2009, pp. 1121–1128.
[102] Ying Nian Wu, Zhangzhang Si, Haifeng Gong, and Song-Chun Zhu, *Learning active basis model for object detection and recognitio*, International Journal of Computer Vision **90** (2010), 198–235.
[103] Ying Nian Wu, Song Chun Zhu, and Xiuwen Liu, *Equivalence of julesz ensembles and frame models*, International Journal of Computer Vision **38** (2000), no. 3, 247–265.
[104] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu, *Cooperative learning of energy-based model and latent variable model via mcmc teaching*, AAAI, 2018.
[105] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu, *Cooperative training of descriptor and generator networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2018), no. preprints.
[106] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu, *A theory of generative convnet*, International Conference on Machine Learning, 2016, pp. 2635–2644.
[107] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu, *Synthesizing dynamic patterns by spatial-temporal generative convnet*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7093–7101.
[108] Laurent Younes, *On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates*, Stochastics: An International Journal of Probability and Stochastic Processes **65** (1999), no. 3-4, 177–228.
[109] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao, *LSUN: construction of a large-scale image dataset using deep learning with humans in the loop*, CoRR **abs/1506.03365** (2015).
[110] Matthew D Zeiler, Graham W Taylor, and Rob Fergus, *Adaptive deconvolutional networks for mid and high level feature learning*, International Conference on Computer Vision, 2011, pp. 2018–2025.
[111] Junbo Zhao, Michael Mathieu, and Yann LeCun, *Energy-based generative adversarial network*, arXiv preprint arXiv:1609.03126 (2016).
[112] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva, *Learning deep features for scene recognition using places database*, Advances in Neural Information Processing Systems, 2014, pp. 487–495.
[113] Song-Chun Zhu, *Statistical modeling and conceptualization of visual patterns*, IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003), no. 6, 691–712.

[114] Song-Chun Zhu, Cheng-En Guo, Yizhou Wang, and Zijian Xu, *What are textons?*, International Journal of Computer Vision **62** (2005), no. 1-2, 121–143.

[115] Song-Chun Zhu and David Mumford, *Grade: Gibbs reaction and diffusion equations.*, International Conference on Computer Vision, 1998, pp. 847–854.

[116] Song-Chun Zhu, David Mumford, et al., *A stochastic grammar of images*, Foundations and Trends® in Computer Graphics and Vision **2** (2007), no. 4, 259–362.

[117] Song-Chun Zhu, Ying Nian Wu, and David Mumford, *Minimax entropy principle and its application to texture modeling*, Neural Computation **9** (1997), no. 8, 1627–1660.