Modeling Natural Images Using Gated MRFs

Marc'Aurelio Ranzato, Volodymyr Mnih, Joshua M. Susskind, Geoffrey E. Hinton

Abstract—This paper describes a Markov Random Field for real-valued image modeling that has two sets of latent variables. One set is used to gate the interactions between all pairs of pixels while the second set determines the mean intensities of each pixel. This is a powerful model with a conditional distribution over the input that is Gaussian with both mean and covariance determined by the configuration of latent variables, which is unlike previous models that were restricted to use Gaussians with either a fixed mean or a diagonal covariance matrix. Thanks to the increased flexibility, this gated MRF can generate more realistic samples after training on an unconstrained distribution of high-resolution natural images. Furthermore, the latent variables of the model can be inferred efficiently and can be used as very effective descriptors in recognition tasks. Both generation and discrimination drastically improve as layers of binary latent variables are added to the model, yielding a hierarchical model called a Deep Belief Network.

Index Terms—gated MRF, natural images, deep learning, unsupervised learning, density estimation, energy-based model, Boltzmann machine, factored 3-way model, generative model, object recognition, denoising, facial expression recognition

1 INTRODUCTION

THE study of the statistical properties of natural images has a long history and has influenced many fields, from image processing to computational neuroscience [1]. In computer vision, for instance, ideas and principles derived from image statistics and from studying the processing stages of the human visual system have had a significant impact on the design of descriptors that are useful for discrimination. A common paradigm has emerged over the past few years in object and scene recognition systems. Most methods [2] start by applying some well-engineered features, like SIFT [3], HoG [4], SURF [5], or PHoG [6], to describe image patches, and then aggregating these features at different spatial resolutions and on different parts of the image to produce a feature vector which is subsequently fed into a general purpose classifier, such as a Support Vector Machine (SVM). Although very successful, these methods rely heavily on human design of good patch descriptors and ways to aggregate them. Given the large and growing amount of easily available image data and continued advances in machine learning, it should be possible to exploit the statistical properties of natural images more efficiently by *learning* better patch descriptors and better ways of aggregating them. This will be particularly significant for data where human expertise is limited such as microscopic, radiographic or hyper-spectral imagery.

In this paper, we focus on probabilistic models of natural images which are useful not only for extracting representations that can subsequently be used for discriminative tasks [7], [8], [9], but also for providing adaptive priors that can be used for image restoration tasks [10], [11], [12]. Thanks to their generative ability, probabilistic models can cope more naturally with ambiguities in the sensory inputs and have the potential to produce more robust features. Devising good models of natural images, however, is a challenging task [1], [12], [13], because images are continuous, high-dimensional and very highly structured.

Recent studies have tried to capture high-order dependencies by using hierarchical models that extract highly non-linear representations of the input [14], [15]. In particular, *deep learning* methods construct hierarchies composed of multiple layers by greedily training each layer separately using unsupervised algorithms [8], [16], [17], [18]. These methods are appealing because 1) they adapt to the input data; 2) they recursively build hierarchies using unsupervised algorithms, breaking up the difficult problem of learning hierarchical non-linear systems into a sequence of simpler learning tasks that use only unlabeled data; 3) they have demonstrated good performance on a variety of domains, from generic object recognition to action recognition in video sequences [17], [18], [19].

In this paper we propose a probabilistic generative model of images that can be used as the front-end of a standard deep architecture, called a Deep Belief Network (DBN) [20]. We test both the generative ability of this model and the usefulness of the representations that it learns for applications such as object recognition, facial expression recognition and image denoising, and we demonstrate state-of-the-art performance for several different tasks involving several different types of image.

Our probabilistic model is called a *gated* Markov Random Field (MRF) because it uses one of its two sets of latent variables to create an image-specific energy function that models the covariance structure of the pixels by switching in sets of pairwise interactions. It uses its other set of latent variables to model the intensities of the pixels [13]. The DBN then uses several further layers of Bernoulli

- *M. Ranzato, V. Mnih and G.E. Hinton are with the Department of Computer Science, University of Toronto, Toronto, ON, M5S 3G4, CANADA.*
  E-mail: see http://www.cs.toronto.edu/˜ranzato
- *J.M. Susskind is with Machine Perception Laboratory, University of California San Diego, La Jolla, 92093, U.S.A.*

latent variables to model the statistical structure in the hidden activities of the two sets of latent variables of the gated MRF. By replicating features in the lower layers it is possible to learn a very good generative model of high-resolution images and to use this as a principled framework for learning adaptive descriptors that turn out to be very useful for discriminative tasks.

In the reminder of this paper, we first discuss our new contributions with respect to our previous published work and then describe the model in detail. In sec. 2 we review other popular generative models of images and motivate the need for the model we propose, the gated MRF. In sec. 3, we describe the learning algorithm as well as the inference procedure for the gated MRF. In order to capture the dependencies between the latent variables of the gated MRF, several other layers of latent variables can be added, yielding a DBN with many layers, as described in sec. 4. Such models cannot be scaled in a simple way to deal with high-resolution images because the number of parameters scales quadratically with the dimensionality of the input at each layer. Therefore, in sec. 5 an efficient and effective weight-sharing scheme is introduced. The key idea is to replicate parameters across local neighborhoods that do not overlap in order to accomplish a twofold goal: exploit stationarity of images while limiting the redundancy of latent variables encoding features at nearby image locations. Finally, we present a thorough validation of the model in sec. 6 with comparisons to other models on a variety of image types and tasks.

1.1 Contributions

This paper is a coherent synthesis of previously unpublished results with the authors' previous work on gated MRFs [21], [9], [13], [22] that has appeared in several recent conference papers and is intended to serve as the main reference on the topic, describing in a more organized and consistent way the major ideas behind this probabilistic model, clarifying the relationship between the mPoT and mcRBM models described below, and providing more details (including pseudo-code) about the learning algorithms and the experimental evaluations. We have included a subsection on the relation to other classical probabilistic models that should help the reader better understand the advantages of the gated MRF and the similarities to other well-known models. The paper includes empirical evaluations of the model on an unusually large variety of tasks, not only on an image denoising and generation tasks that are standard ways to evaluate probabilistic generative models of natural images, but also on three very different recognition tasks (scenes, generic object recognition, and facial expressions under occlusion). The paper demonstrates that the gated MRF can be used for a wide range of different vision tasks, and it should suggest many other tasks that can benefit from the generative power of the model.

2 The Gated MRF

In this section, we first review some of the most popular probabilistic models of images and discuss how their underlying assumptions limit their modeling abilities. This motivates the introduction of the model we propose. After describing our basic model and its learning and inference procedures, we show how we can make it hierarchical and how we can scale it up using parameter-sharing to deal with high-resolution images.

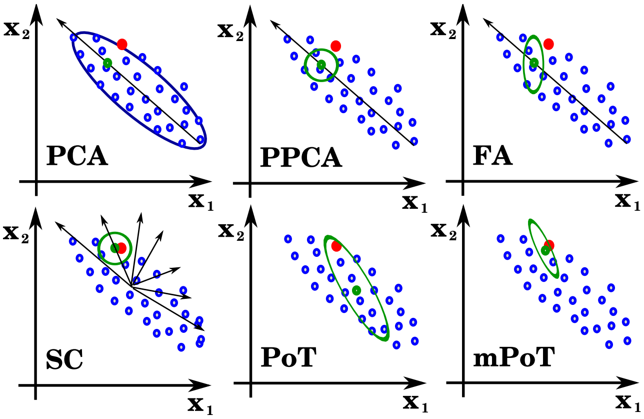

Fig. 1. Toy illustration to compare different models. x-axis is the first pixel, y-axis is the second pixel of two-pixel images. Blue dots are a dataset of two-pixel images. The red dot is the data point we want to represent. The green dot is its (mean) reconstruction. The models are: Principal Component Analysis, Probabilistic PCA, Factor Analysis, Sparse Coding, Product of Student's t and mean PoT.

2.1 Relation to Other Probabilistic Models

Natural images live in a very high dimensional space that has as many dimensions as number of pixels, easily in the order of millions and more. Yet it is believed that they occupy a tiny fraction of that space, due to the structure of the world, encompassing a much lower dimensional yet highly non-linear manifold [23]. The ultimate goal of unsupervised learning is to discover representations that parameterize such a manifold, and hence, capture the intrinsic structure of the input data. This structure is represented through features, also called latent variables in probabilistic models.

One simple way to check whether a model extracts features that retain information about the input, is by reconstructing the input itself from the features. If reconstruction errors of inputs similar to training samples is lower than reconstruction errors of other input data points, then the model must have learned interesting regularities [24]. In PCA, for instance, the mapping into feature space is a linear projection into the leading principal components and the reconstruction is performed by another linear projection. The reconstruction is perfect only for those data points that lie in the linear subspace spanned by the leading principal components. The principal components are the structure captured by this model.

Also in a probabilistic framework we have a mapping into feature, or latent variable, space and back to image space. The former is obtained by using the *posterior* distribution over the latent variables, $p(\mathbf{h}|\mathbf{x})$ where $\mathbf{x}$ is the input and $\mathbf{h}$ the latent variables, the latter through the conditional distribution over the input, $p(\mathbf{x}|\mathbf{h})$.

As in PCA one would reconstruct the input from the features in order to assess the quality of the encoding, while in a probabilistic setting we can analyze and compare different models in terms of their conditional $p(\mathbf{x}|\mathbf{h})$. We can sample the latent variables, $\bar{\mathbf{h}} \sim p(\mathbf{h}|\bar{\mathbf{x}})$ given an input image $\bar{\mathbf{x}}$, and then look at how well the image $\bar{\mathbf{x}}$ can be reconstructed using $p(\mathbf{x}|\bar{\mathbf{h}})$. Reconstructions produced in this way are typically much more like real data than true samples from the underlying generative model because the latent variables are sampled from their posterior distribution, $p(\mathbf{h}|\bar{\mathbf{x}})$, rather than from their prior, $p(\mathbf{h})$, but the reconstructions do provide insight into how much of the information in the image is preserved in the sampled values of the latent variables.

As shown in fig. 1, most models such as Probabilistic Principal Component Analysis (PPCA) [25], Factor Analysis (FA) [26], Independent Component Analysis (ICA) [27], Sparse Coding (SC) [28], and Gaussian Restricted Boltzmann Machines (GRBM) [29], assume that the conditional distribution of the pixels $p(\mathbf{x}|\mathbf{h})$ is Gaussian with a mean determined by the latent variables and a fixed, image-independent covariance matrix. In PPCA the mean of the distribution lies along the directions of the leading eigenvectors while in SC it is along a linear combination of a very small number of basis vectors (represented by black arrows in the figure). From a generative point of view, these are rather poor assumptions for modeling natural images because much of the interesting structure of natural images lies in the fact that the covariance structure of the pixels varies considerably from image to image. A vertical occluding edge, for example, eliminates the typical strong correlation between pixels on opposite sides of the edge.

This limitation is addressed by models like Product of Student's t (PoT) [30], covariance Restricted Boltzmann Machine (cRBM) [21] and the model proposed by Karklin and Lewicki [14] each of which instead assume a Gaussian conditional distribution with a fixed mean but with a full covariance determined by the states of the latent variables. Latent variables explicitly account for the correlation patterns of the input pixels, avoiding interpolation across edges while smoothing within uniform regions. The mean, however, is fixed to the average of the input data vectors across the whole dataset. As shown in the next section, this can yield very poor conditional models of the input distribution.

In this work, we extend these two classes of models with a new model whose conditional distribution over the input has both a mean and a covariance matrix determined by latent variables. We will introduce two such models, namely the mean PoT (mPoT) [13] and the mean-covariance RBM (mcRBM) [9], which differ only in the choice of their distribution over latent variables. We refer to these models as *gated MRF*'s because they are pair-wise Markov Random Fields (MRFs) with latent variables gating the couplings between input variables. Their marginal distribution can be interpreted as a mixture of Gaussians with an infinite (mPoT) or exponential (mcRBM) number of components, each with non-zero mean and full covariance matrix and tied parameters.

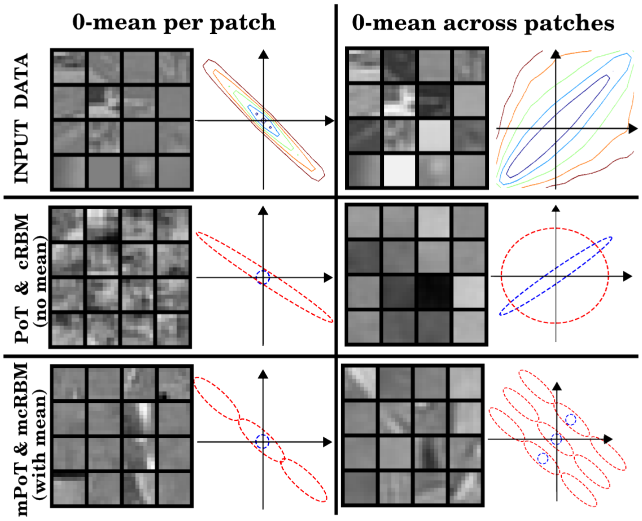

Fig. 2. In the first column, each image is zero mean. In the second column, the whole data set is centered but each image can have non-zero mean. First row: 8x8 natural image patches and contours of the empirical distribution of (tiny) two-pixel images (the x-axis being the first pixel and the y-axis the second pixel). Second row: images generated by a model that does not account for mean intensity with plots of how such model could fit the distribution of two-pixel images using mixture of Gaussians with components that can choose between two covariances. Third row: images generated by a model that has both "mean" and "covariance" hidden units and toy-illustration of how such model can fit the distribution of two-pixel images discovering the manifold of structured images (along the anti-diagonal) using a mixture of Gaussians with arbitrary mean and only two covariances.

2.2 Motivation

A Product of Student's t (PoT) model [31] can be viewed as modelling image-specific, pair-wise relationships between pixel values by using the states of its latent variables. It is very good at representing the fact that two pixels have very similar intensities and no good at all at modelling what these intensities are. Failure to model the mean also leads to impoverished modelling of the covariances when the input images have non-zero mean intensity. The covariance RBM (cRBM) [21] is another model that shares the same limitation since it only differs from PoT in the distribution of its latent variables: The posterior over the latent variables $p(\mathbf{h}|\mathbf{x})$ is a product of Bernoulli distributions instead of Gamma distributions as in PoT.

We explain the fundamental limitation of these models by using a simple toy example: Modelling two-pixel images using a cRBM with only one binary latent variable (see fig. 2). This cRBM assumes that the conditional distribution over the input $p(\mathbf{x}|\mathbf{h})$ is a zero-mean Gaussian with a covariance that is determined by the state of the latent variable. Since the latent variable is binary, the cRBM can be viewed as a mixture of two zero-mean full covariance Gaussians. The latent variable uses the pairwise relationship between pixels to decide which of the two covariance matrices should be used to model each image. When the input data is pre-processed by making each image have zero mean intensity (the plot of the empirical histogram is shown in the first row and first column), most images lie near the origin because most of the times nearby pixels are strongly correlated. Less frequently we encounter edge images that exhibit strong anti-correlation between the pixels, as shown by the long tails along the anti-diagonal line. A cRBM could model this data by using two Gaussians (second row and first column): one that is spherical and tight at the origin for smooth images and another one that has a covariance elongated along the anti-diagonal for structured images.

If, however, the whole set of images is normalized by subtracting from every pixel the mean value of all pixels over all images (first row and second column), the cRBM fails at modelling structured images (second row and second column). It can fit a Gaussian to the smooth images by discovering the direction of strong correlation along the main diagonal, but it is very likely to fail to discover the direction of anti-correlation, which is crucial to represent discontinuities, because structured images with different mean intensity appear to be evenly spread over the whole input space.

If the model has another set of latent variables that can change the means of the Gaussian distributions in the mixture (as explained more formally below and yielding the mPoT and mcRBM models), then the model can represent both changes of mean intensity and the correlational structure of pixels (see last row). The mean latent variables effectively subtract off the relevant mean from each datapoint, letting the covariance latent variable capture the covariance structure of the data. As before, the covariance latent variable needs only to select between two covariance matrices.

In fact, experiments on real 8x8 image patches confirm these conjectures. Fig. 2 shows samples drawn from PoT and mPoT. The mPoT model (and similarly mcRBM [9]) is better at modelling zero mean images and much better at modelling images that have non-zero mean intensity. This will be particularly relevant when we introduce a convolutional extension of the model to represent spatially stationary high-resolution images (as opposed to small image patches), since it will not be possible to independently normalize overlapping image patches.

As we shall see in sec. 6.1, models that do not account for mean intensity cannot generate realistic samples of natural images since samples drawn from the conditional distribution over the input have expected intensity that is constant everywhere regardless of the value of the latent variables. In the model we propose instead there is a set of latent variables whose role is to bias the average mean intensity differently in different regions of the input image. Combined with the correlational structure provided by the covariance latent variables, this produces smooth images that have sharp boundaries between regions of different mean intensity.

2.3 Energy Functions

We start the discussion assuming the input is a small vectorized image patch, denoted by $\mathbf{x} \in \mathbb{R}^D$, and the latent variables are denoted by the vector $\mathbf{h}^p \in \{0,1\}^N$. First, we consider a pair-wise MRF defined in terms of an energy function E. The probability density function is related to E by: $p(\mathbf{x}, \mathbf{h}^p) = \exp(-E(\mathbf{x}, \mathbf{h}^p))/Z$, where Z is an (intractable) normalization constant which is called the *partition function*. The energy is:

$$E(\mathbf{x}, \mathbf{h}^p) = \frac{1}{2} \sum_{i,j,k} t_{ijk} x_i x_j h^p_k \tag{1}$$

The states of the latent variables, called *precision* hidden units, modulate the pair-wise interactions t_{ijk} between all pairs of input variables x_i and x_j, with $i, j = 1..D$. Similarly to Sejnowski [32], the energy function is defined in terms of *3-way multiplicative interactions*. Unlike previous work by Memisevic and Hinton [33] on modeling image transformations, here we use this energy function to model the *joint* distribution of the variables within the vector $\mathbf{x}$.

This way of allowing hidden units to modulate interactions between input units has far too many parameters. For real images we expect the required lateral interactions to have a lot of regular structure. A hidden unit that represents a vertical occluding edge, for example, needs to modulate the lateral interactions so as to eliminate horizontal interpolation of intensities in the region of the edge. This regular structure can be approximated by writing the 3-dimensional tensor of parameters t as a sum of outer products: $t_{ijk} = \sum_f C^{(1)}_{if} C^{(2)}_{jf} P_{fk}$, where f is an index over F deterministic factors, $C^{(1)}$ and $C^{(2)} \in \mathbb{R}^{D \times F}$, and $P \in \mathbb{R}^{F \times N}$. Since the factors are connected twice to the same image through matrices $C^{(1)}$ and $C^{(2)}$, it is natural to tie their weights further reducing the number of parameters, yielding the final parameterization $t_{ijk} = \sum_f C_{if} C_{jf} P_{fk}$. Thus, taking into account also the hidden biases, eq. 1 becomes:

$$E(\mathbf{x}, \mathbf{h}^p) = \frac{1}{2} \sum_{f=1}^{F} (\sum_{k=1}^{N} P_{fk} h^p_k)(\sum_{i=1}^{D} C_{if} x_i)^2 - \sum_{k=1}^{N} b^p_k h^p_k \tag{2}$$

which can be written more compactly in matrix form as:

$$E(\mathbf{x}, \mathbf{h}^p) = \frac{1}{2} \mathbf{x}^{\mathrm{T}} C \mathrm{diag}(P \mathbf{h}^p) C^{\mathrm{T}} \mathbf{x} - \mathbf{b}^{p\mathrm{T}} \mathbf{h}^p \tag{3}$$

where $\mathrm{diag}(\mathbf{v})$ is a diagonal matrix with diagonal entries given by the elements of vector $\mathbf{v}$. This model can be interpreted as an instance of an RBM modeling pair-wise interactions between the input pixels[1] and we dub it covariance RBM (cRBM) [21], [9][2] since it models the covariance structure of the input through the "precision" latent variables $\mathbf{h}^p$.

The hidden units remain conditionally independent given the states of the input units and their binary states can be sampled using:

$$p(h^p_k = 1 | \mathbf{x}) = \sigma\Big(-\frac{1}{2} \sum_{f=1}^{F} P_{fk} (\sum_{i=1}^{D} C_{if} x_i)^2 + b^p_k \Big) \tag{4}$$

[1] More precisely, this is an instance of a semi-restricted Boltzmann machine [34], [35], since only hidden units are "restricted", i.e. lack lateral interactions.

[2] This model should not be confused with the conditional RBM [36].

where σ is the logistic function $\sigma(v) = 1/\big(1+\exp(-v)\big)$. Given the states of the hidden units, the input units form an MRF in which the effective pairwise interaction weight between x_i and x_j is $\frac{1}{2}\sum_f \sum_k P_{fk} h_k^p C_{if} C_{jf}$. Therefore, the conditional distribution over the input is:

$$p(\mathbf{x}|\mathbf{h}^p) = N(0, \Sigma), \text{ with } \Sigma^{-1} = C\text{diag}(P\mathbf{h}^p)C^{\mathrm{T}} \quad (5)$$

Notice that the covariance matrix is not fixed, but is a function of the states of the precision latent variables $\mathbf{h}^p$. In order to guarantee positive definiteness of the covariance matrix we need to constrain P to be *non-negative* and add a small quadratic regularization term to the energy function[3], here ignored for clarity of presentation.

As described in sec. 2.2, we want the conditional distribution over the pixels to be a Gaussian with not only its covariance but also its mean depending on the states of the latent variables. Since the product of a full covariance Gaussian (like the one in eq. 5) with a spherical non-zero mean Gaussian is a non-zero mean full covariance Gaussian, we simply *add* the energy function of cRBM in eq. 3 to the energy function of a GRBM [29], yielding:

$$E(\mathbf{x}, \mathbf{h}^m, \mathbf{h}^p) = \frac{1}{2}\mathbf{x}^{\mathrm{T}} C\text{diag}(P\mathbf{h}^p)C^{\mathrm{T}}\mathbf{x} - \mathbf{b}^{p\mathrm{T}}\mathbf{h}^p$$
$$+\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x} - \mathbf{h}^m W^{\mathrm{T}}\mathbf{x} - \mathbf{b}^{m\mathrm{T}}\mathbf{h}^m - \mathbf{b}^{x\mathrm{T}}\mathbf{x} \quad (6)$$

where $\mathbf{h}^m \in \{0,1\}^M$ are called "mean" latent variables because they contribute to control the mean of the conditional distribution over the input:

$$p(\mathbf{x}|\mathbf{h}^m, \mathbf{h}^p) = N\Bigg(\Sigma(W\mathbf{h}^m + \mathbf{b}^x), \Sigma\Bigg), \quad (7)$$
$$\text{with } \Sigma^{-1} = C\text{diag}(P\mathbf{h}^p)C^{\mathrm{T}} + I$$

where I is the identity matrix, $W \in \mathbb{R}^{D\times M}$ is a matrix of trainable parameters and $\mathbf{b}^x \in \mathbb{R}^D$ is a vector of trainable biases for the input variables. The posterior distribution over the mean latent variables is[4]:

$$p(h_k^m = 1|\mathbf{x}) = \sigma(\sum_{i=1}^{D} W_{ik}x_i + b_k^m) \quad (8)$$

The overall model, whose joint probability density function is proportional to $\exp(-E(\mathbf{x}, \mathbf{h}^m, \mathbf{h}^p))$, is called a mean covariance RBM (mcRBM) [9] and is represented in fig. 3.

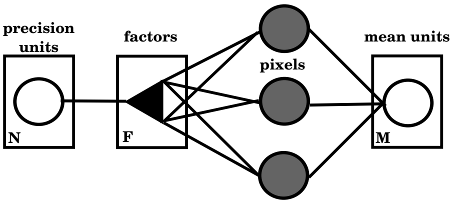

Fig. 3. Graphical model representation (with only three input variables): There are two sets of latent variables (the mean and the precision units) that are conditionally independent given the input pixels and a set of deterministic factor nodes that connect triplets of variables (pairs of input variables and one precision unit).

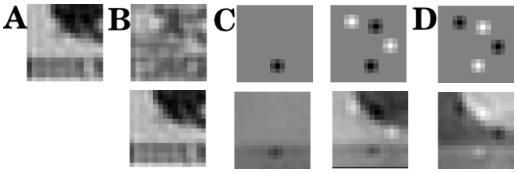

Fig. 4. A) Input image patch. B) Reconstruction performed using only mean hiddens (i.e. $W\mathbf{h}^m + \mathbf{b}^x$) (top) and both mean and precision hiddens (bottom) (that is multiplying the patch on the top by the image-specific covariance $\Sigma = \big(C\text{diag}(P\mathbf{h}^p)C^{\mathrm{T}} + I\big)^{-1}$, see mean of Gaussian in eq. 7). C) Reconstructions produced by combining the correct image-specific covariance as above with the incorrect, hand-specified pixel intensities shown in the top row. Knowledge about pair-wise dependencies allows a blob of high or low intensity to be spread out over the appropriate region. D) Reconstructions produced like in C) showing that precision hiddens do not account for polarity (nor for the exact intensity values of regions) but only for correlations.

The demonstration in fig. 4 is designed to illustrate how the mean and precision latent variables cooperate to represent the input. Through the precision latent variables the model knows about pair-wise correlations in the image. For instance, it knows that the pixels in the lower part of the image in fig. 4-A are strongly correlated; these pixels are likely to take the same value, but the precision latent variables do *not* carry any information about which value this is. Then, very noisy information about the values of the individual pixels in the lower part of the image, as those provided by the mean latent variables, would be sufficient to reconstruct the whole region quite well, since the model knows which values can be smoothed. Mathematically, the interaction between mean and precision latent variables is expressed by the product between Σ (which depends only on $\mathbf{h}^p$) and $W\mathbf{h}^m + \mathbf{b}^x$ in the mean of the Gaussian distribution of eq. 7. We can repeat the same argument for the pixels in the top right corner and for those in the middle part of the image as well. Fig. 4-C illustrates this concept, while fig. 4-D shows that flipping the sign of the reconstruction of the mean latent variables flips the sign of the overall reconstruction as expected. Information about intensity is propagated over each region thanks to the pair-wise dependencies captured by the precision hidden units. Fig. 4-B shows the same using the *actual* mean intensity produced by the mean hidden units (top). The reconstruction produced by the model using the whole set of hidden units is very close to the input image, as can be seen in the bottom part of fig. 4-B.

In a mcRBM the posterior distribution over the latent variables $p(\mathbf{h}|\mathbf{x})$ is a product of Bernoullis as shown in eq. 8 and 4. This distribution is particularly easy to use in a standard DBN [20] where each layer is trained using a binary-binary RBM. Binary latent variables, however, are not very good at representing different real-values of

[3] In practice, this term is not needed when the dimensionality of $\mathbf{h}^p$ is larger than $\mathbf{x}$.

[4] Notice how the mean latent variables compute a non-linear projection of a linear filter bank, akin to the most simplified "simple-cell" model of area V1 of the visual cortex, while the precision units perform an operation similar to the "complex-cell" model because rectified (squared) filter outputs are non-linearly pooled to produce their response. In this model, simple and complex cells perform their operations in parallel (not sequentially). If, however, we equate the factors used by the precision units to simple cells, we recover the standard model in which simple cells send their squared outputs to complex cells.

the mean intensity or different levels of contrast across an edge. A binary latent variable can represent a probability of 0.71 that a binary feature is present, but this is not at all the same as representing that a real-valued intensity is 0.71 and definitely not 0.70 or 0.72. In our previous work [21] we showed that by combining many binary variables with shared weights and offset biases we can closely approximate continuous Gamma units, however. This issue can be addressed in several other ways, by either normalizing the input data $\mathbf{x}$, or by normalizing the input in the energy function [9], or by changing the distribution of the latent variables.

In our experiments, we tried a few distributions[5] and found that Bernoulli variables for the mean hiddens and Gamma variables for the precision hiddens gave the best generative model. The resulting model, dubbed mean PoT (mPoT), is a slight modification of mcRBM. The energy is:

$$E(\mathbf{x}, \mathbf{h}^m, \mathbf{h}^p) = \frac{1}{2}\mathbf{x}^{\mathrm{T}} C \mathrm{diag}(P\mathbf{h}^p) C^{\mathrm{T}}\mathbf{x} + \mathbf{1}^{\mathrm{T}}(\mathbf{h}^p + (1-\gamma)\log \mathbf{h}^p) + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x} - \mathbf{h}^m W^{\mathrm{T}}\mathbf{x} - \mathbf{b}^{m\mathrm{T}}\mathbf{h}^m - \mathbf{b}^{x\mathrm{T}}\mathbf{x} \quad (9)$$

where $\mathbf{1}$ is a vector of 1's, $\gamma \in \mathbb{R}^+$ and $\mathbf{h}^p$ is a positive vector of real valued variables [31]. In mPoT the posterior distribution over the precision hiddens is:

$$p(h_j^p|\mathbf{x}) = \Gamma(\gamma, 1 + \frac{1}{2}\sum_f P_{fj} (\sum_i C_{if} x_i)^2) \qquad (10)$$

where Γ is the Gamma distribution with expected value:

$$E[h_j^p|\mathbf{x}] = \frac{\gamma}{1 + \frac{1}{2}\sum_f P_{fj}(\sum_i C_{if} x_i)^2} \qquad (11)$$

Compared to eq. 4, we see that the operations required to compute the expectations are very similar, except for the non-linear transformation on the pooled squared filter outputs.

2.4 Relation to Line Processes and Sparse Coding

Let us consider the energy function of eq. 2 assuming that P is set to identity and the biases are all large and positive. The energy penalizes those images $\mathbf{x}$ that yield large filter responses (because of the positive sign of the energy function) [37]. Without latent variables that have the possibility of turning off, the model would discover the *minor components* of the data [38], i.e. the filter weights would represent the directions of the minor components or linear combinations of those directions. The introduction of latent variables makes the model highly non-linear, causing it to learn filters with heavy-tailed output distributions. In the rare event of a non-smooth image (e.g., an edge in a certain location and orientation), a large filter output will cause the corresponding latent variable to turn off so that the increase in energy is equal to the bias rather than quadratic in the filter output value. This leads to *sparse* filter outputs that are usually close to zero but occasionally much larger. By default hidden units are all "on" because the input is not typically aligned with the oriented filters, but when an input $\mathbf{x}$ matches a filter, the corresponding hidden unit turns "off" thus representing the violation of a smoothness constraint.

In other words, images are assumed to be almost always smooth, but rare violations of this constraint are allowed by using auxiliary (latent) variables that act like switches. This idea was first proposed by Geman and Geman [39] and later revisited by many others [40], [41]. Like the Gaussian Scale Mixture model [10], our model uses hidden variables that control the modeled covariance between pixels, but our inference process is simpler because the model is undirected. Also, all parameters of our model are learned.

2.5 Notes on the Modelling Choice

As stated in sec. 2.2, the main motivation for the model we propose is to define a conditional distribution over the input $p(\mathbf{x}|\mathbf{h}^m, \mathbf{h}^p)$ which is a multivariate Gaussian with both mean and covariance determined by the state of latent variables. In order to achieve this, we have defined an energy function of the type: $E = \frac{1}{2}\mathbf{x}^T \Sigma_{\mathbf{h}^p}^{-1}\mathbf{x} + W_{\mathbf{h}^m}\mathbf{x}$, where we denote with $\Sigma_{\mathbf{h}^p}$ and $W_{\mathbf{h}^m}$ matrices that depend on $\mathbf{h}^p$ and $\mathbf{h}^m$, respectively. Two questions naturally arise. First, would the model work as well if we tie the set of latent variables that control the mean and covariance? And second, are there other formulations of energy functions that define multivariate Gaussian conditional distributions?

The answer to the first question is negative for a model of natural images since the statistics of the covariance structure is heavy tailed (nearby pixels are almost always strongly correlated) while the statistics of the mean intensity is not. This affects the statistics of the mean and covariance latent variables as well. The former is typically much sparser than the latter, therefore, tying both sets of latent variables would constrain the model in an unnatural way.

The answer to the second question is positive instead. The alternative would be the following energy function: $E = \frac{1}{2}(\mathbf{x} - W_{\mathbf{h}^m})^T \Sigma_{\mathbf{h}^p}^{-1}(\mathbf{x} - W_{\mathbf{h}^m})$. The advantage of this formulation is that the parameters defining the mean are defined in the image domain and, therefore, are much easier to interpret. The mean parameters directly define the mean of the Gaussian distribution, while in our formulation the mean of the Gaussian distribution is given by a product, $\Sigma_{\mathbf{h}^p} W_{\mathbf{h}^m}$, see eq. 7. The fundamental disadvantage of this formulation is that inference of the latent variables becomes inefficient since the energy function has multiplicative interactions between latent variables which make them conditionally *dependent* given the input. Lengthy iterative procedures would be required to compute a sample from $p(\mathbf{h}^m|\mathbf{x})$ and $p(\mathbf{h}^p|\mathbf{x})$.

3 Learning Algorithm

The models described in the previous section can be reformulated by integrating out the latent variables over their

[5] An extensive evaluation of different choices of latent variable distributions is not explored here, and it is subject of future study.

domain. In the case of mPoT, for instance, we have that the free energy $\mathcal{F}(\mathbf{x}) = -\log \int_{\mathbf{h}^p} \sum_{\mathbf{h}^m} \exp(-E(\mathbf{x}, \mathbf{h}^m, \mathbf{h}^p))$ is:

$$\begin{aligned}\mathcal{F}(\mathbf{x}) \;=\; & \sum_{j=1}^{N} \log(1 + \frac{1}{2} \sum_{f=1}^{F} P_{fj} (\sum_{i=1}^{D} C_{if} x_i)^2) + \sum_{i=1}^{D} \frac{1}{2} x_i^2 \\ & - \sum_{i=1}^{D} b_i^x x_i - \sum_{k=1}^{M} \log(1 + e^{\sum_i W_{ik} x_i + b_k^m}) \quad (12)\end{aligned}$$

making the marginal distribution, $p(\mathbf{x}) \propto \exp(-\mathcal{F}(\mathbf{x}))$, a product of experts [42].

Let us denote a generic parameter of the model with $\theta \in \{C, P, \gamma, W, \mathbf{b}^x, \mathbf{b}^m\}$. We learn the parameters by stochastic gradient ascent in the log likelihood. We can write the likelihood in terms of the joint energy E or in terms of the free energy $\mathcal{F}$. In both cases, maximum likelihood requires us to compute expectations over the model distributions. These can be approximated with Monte Carlo sampling algorithms. When using E the most natural sampling algorithm is Gibbs sampling (alternating the sampling from $p(\mathbf{h}^m|\mathbf{x})$ and $p(\mathbf{h}^p|\mathbf{x})$ to $p(\mathbf{x}|\mathbf{h}^m, \mathbf{h}^p)$), while Hybrid Monte Carlo (HMC) [43] is a better sampling algorithm when using $\mathcal{F}$ since $\mathbf{x} \in \mathbb{R}^D$ and $\mathcal{F}$ is differentiable. It is beyond the scope of this work to analyze and compare different sampling methods. Our preliminary experiments showed that training with HMC yielded models that produce better visually looking samples, and therefore, we will now describe the algorithm in terms $\mathcal{F}$ only.

The update rule for gradient ascent in the likelihood is:

$$\theta \leftarrow \theta + \eta \left(< \frac{\partial \mathcal{F}}{\partial \theta} >_{\text{model}} - < \frac{\partial \mathcal{F}}{\partial \theta} >_{\text{data}} \right) \quad (13)$$

where $<>$ denotes expectation over samples from the model or the training data. While it is straightforward to compute the value and the gradient of the free energy with respect to θ, computing the expectations over the model distribution is intractable because we would need to run the HMC sampler for a very long time, discarding and reinitializing the momentum auxiliary variables many times [43]. We approximate that with Fast Persistent Contrastive Divergence (FPCD) [44]. We run the sampler for only one step[6] starting at the previous sample drawn from the model and using as parameters the sum of the original parameters and a small perturbation vector which adapts rapidly but also decays towards zero rapidly. The perturbation vector repels the system from its current state by raising the energy of that state and it therefore encourages samples to explore the input space rapidly even when the learning rate for the model parameters is very small (see [44] for more details). The learning algorithm loops over batches of training samples and: (1) it computes $< \frac{\partial \mathcal{F}}{\partial \theta} >$ at the training samples, (2) it generates "negative" samples by running HMC for just one set of 20 leap-frog steps (using the slightly perturbed parameter vector in the free energy function), (3) it computes $< \frac{\partial \mathcal{F}}{\partial \theta} >$ at the negative samples, and (4) it updates the parameters using eq. 13 (see algorithm in fig. 5).

```
def train_DBN():
 # loop from bottom to top layer
 for k in range(number_of_layers):
  W[k] = random_init()
  Wf[k] = 0 # fast weights used by FPCD
  neg_batch = random_init() # init. samples from model
  # sweep over data divided into minibatches
  for epoch in range(number_of_epochs):
   for batch in range(number_of_minibatches):
    g1 = compute_dFdW(batch,W[k])
    neg_batch = draw_sample(neg_batch,W[k]+Wf[k],k)
    g2 = compute_dFdW(neg_batch,W[k]+Wf[k])
    W[k] = W[k] - (learn_rate/epoch)*(g1-g2 + decay*W[k])
    Wf[k] = 19/20*Wf[k] - learn_rate*(g1-g2)
  # make minibatches for layer above by computing E[h|x]
  generate_data_using_posterior(batches,W[k])


def draw_sample(datainit,param,layer): # only 1 M.C. step
 if layer == 1: # 1st layer: do 1 step of HMC
  velocity = randn()
  tot_energy1 = .5*velocity^2 + compute_F(datainit,param)
  data = datainit
  velocity = velocity - eps * compute_dFdX(data,param)/2
  for iter in range(20): # 20 leap-frog steps
   data = data + eps * velocity
   if iter != 19:
    velocity = velocity - eps * compute_dFdX(data,param)
  velocity = velocity - eps * compute_dFdX(data,param)
  tot_energy2 = .5*velocity^2 + compute_F(data,param)
  if rand() < exp(tot_energy1 - tot_energy2):
   return data # accept sample
  else:
   return datainit # reject sample
 else: # higher layers: do 1 step of Gibbs
  hiddens = sample_posterior(datainit,param) # p(h|x)
  return sample_inputs(hiddens,param) # p(x|h)
```

Fig. 5. Pseudo-code of learning algorithm for DBN using FPCD. Energy function given by eq. 12.

4 Learning a DBN

In their work, Hinton et al. [20] trained DBNs using a greedy layer-wise procedure, proving that this method, if done properly, creates a sequence of lower bounds on the log likelihood of the data, each of which is better than the previous bound. Here, we follow a similar procedure. First, we train a gated MRF to fit the distribution of the input. Then, we use it to compute the expectation of the first layer latent variables conditioned on the input training images. Second, we use these expected values as input to train the second layer of latent variables[7]. Once the second layer is trained, we use it to compute expectations of the second layer latent variables conditioned on the second layer input to provide inputs to the third layer, and so on. The difficult problem of learning a hierarchical model with several layers

[6] One step of HMC is composed of a randomly chosen initial momentum and 20 "leap-frog steps" that follow a dynamical simulation. If the sum of the kinetic and potential energy rises by Δ due to inaccurate simulation of the dynamics, the system is returned to the initial state with probability $1 - \exp(-\Delta)$. The step size of the simulation is adjusted to keep the rejection rate at 10%.

[7] It would be more correct to use stochastically sampled values as the "data", but using the expectations reduces noise in the learning and works almost as well. Also, the second layer RBM expects input values in the interval $[0, 1]$ when using the expectations of Bernoulli variables. Therefore, when we use the precision units of mPoT, we divide the expectation of eq. 11 by γ. A more elegant solution is to change the conditional distribution in the second layer RBM to Gamma. However, the simple rescaling worked well in our experiments.

of latent variables is thus decomposed into a sequence of much simpler learning tasks.

Let us consider the i-th layer of the deep model in isolation. Let us assume that the input to that layer consists of a binary vector denoted by $\mathbf{h}^{i-1} \in \{0,1\}^{N_{i-1}}$. This is modeled by a binary RBM which is also defined in terms of an energy function:

$$E(\mathbf{h}^{i-1}, \mathbf{h}^i) = -\mathbf{h}^{i-1^\mathrm{T}} W^i \mathbf{h}^i - \mathbf{b}^{i-1^\mathrm{T}} \mathbf{h}^{i-1} - \mathbf{b}^{i^\mathrm{T}} \mathbf{h}^i \quad (14)$$

where $W^i \in \mathbb{R}^{N_{i-1} \times N_i}$ is the i-th layer parameter matrix and $\mathbf{b}^i \in \mathbb{R}^{N_i}$ is the i-th layer vector of biases. In a RBM, all input variables are conditionally independent given the latent variables and vice versa; so:

$$p(h_k^i = 1 | \mathbf{h}^{i-1}) = \sigma(\sum_{j=1}^{N_{i-1}} W_{jk}^i h_j^{i-1} + b_k^i), k = 1..N_i \quad (15)$$

$$p(h_j^{i-1} = 1 | \mathbf{h}^i) = \sigma(\sum_{k=1}^{N_i} W_{jk}^i h_k^i + b_j^{i-1}), j = 1..N_{i-1} \quad (16)$$

Therefore, in the higher layers both computing the posterior distribution over the latent variables and computing the conditional distribution over the input variables is very simple and it can be done in parallel.

Learning higher layer RBMs is done using the FPCD algorithm as before, except that HMC is replaced with Gibbs sampling (i.e. samples from the model are updated by sampling from $p(\mathbf{h}^i|\mathbf{h}^{i-1})$ followed by $p(\mathbf{h}^{i-1}|\mathbf{h}^i)$). The pseudo-code of the overall algorithm is given in fig. 5.

Once the model has been trained it can be used to generate samples. The correct sampling procedure [20] consists of generating a sample from the topmost RBM, followed by back-projection to image space through the chain of conditional distributions for each layer given the layer above. For instance, in the model shown in fig. 6 one generates from the top RBM by running a Gibbs sampler that alternates between sampling $\mathbf{h}^2$ and $\mathbf{h}^3$. In order to draw an unbiased sample from the deep model, we then map the second layer sample produced in this way through the conditional distributions $p(\mathbf{h}^m|\mathbf{h}^2)$ and $p(\mathbf{h}^p|\mathbf{h}^2)$ to sample the mean and precision latent variables. These sampled values then determine the mean and covariance of a Gaussian distribution over the pixels, $p(\mathbf{x}|\mathbf{h}^m, \mathbf{h}^p)$.

Given an image, an inexact but fairly accurate way to infer the posterior distribution of the latent variables in the deep model consists of propagating the input through the chain of posterior distributions for each of the greedily learned individual models (i.e. the mPoT or the subsequent RBMs). For each of these models separately, this corresponds to exact inference because, for each separate model, the latent variables are conditionally independent given the "data" variables used for that individual model so no iteration is required. Unfortunately, when the individual models are composed to form a deep belief net, this way of inferring the lower level variables is no longer correct, as explained in [20]. Fortunately, however, the variational bound on the likelihood that is improved as each layer is added assumes this form of incorrect inference so the learning ensures that it works well. Referring to the deep model in fig. 6, we perform inference by computing $p(\mathbf{h}^m|\mathbf{x})$ and $p(\mathbf{h}^p|\mathbf{x})$, followed by $p(\mathbf{h}^2|\mathbf{h}^m, \mathbf{h}^p)$, followed by $p(\mathbf{h}^3|\mathbf{h}^2)$. Notice that all these distributions are factorial and can be computed without any iteration (see eq. 8, 10 and 15).

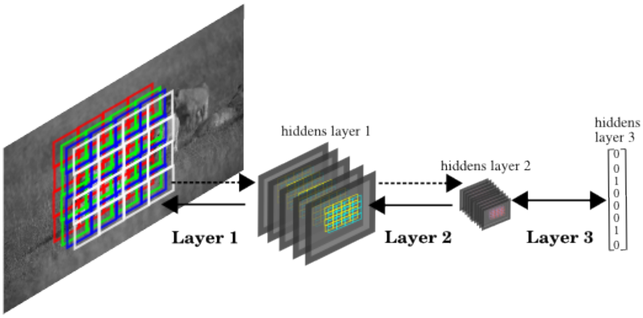

Fig. 6. Outline of a deep generative model composed of three layers. The first layer applies filters that tile the input image with different offsets (squares of different color are filters with different parameters). The filters of this layer are learned on natural images. Afterwards, a second layer is added using as input the expectation of the first layer latent variables. This layer is trained to fit the distribution of its input. The procedure is repeated again for the third layer. After training, inference of the top level representation is well approximated by propagating the expectation of the latent variables given their input starting from the input image (see dashed arrows). Generation is performed by first using a Monte Carlo method to sample from the "layer-3" model, and then using the conditional over the input at each layer given the sample at the layer above to back-project in image space (see continuous line arrows).

5 Scaling to High-Resolution Images

The gated MRF described in the previous sections does not scale well to high-resolution images because each latent variable is connected to all input pixels. Since the number of latent variables scales as the number of input variables, the number of parameters subject to learning scales quadratically with the size of the input making learning infeasibly slow. We can limit the number of parameters by making use of the fact that correlations between pixels usually decay rapidly with distance [45]. In practice, gated MRFs that learn filters on large image patches discover spatially localized patterns. This suggests the use of local filters that connect each latent variable to only a small set of nearby pixels, typically a small square image patch. Both "precision" filters C and "mean" filters W can benefit from *locality*, and P can also be made local by connecting each latent variable to only a small local subset of filter outputs.

In addition to locality, we can exploit *stationarity* to further reduce the number of parameters since (on average) different locations in images have the same statistical properties. This suggests that we should parameterize by replicating each learned filter across all possible locations. This dramatically reduces the number of parameters but, unfortunately, it makes the values of the latent variables highly redundant. Convolutional models [46], [11] typically extract highly overcomplete representations. If k different local filters are replicated over all possible integer positions

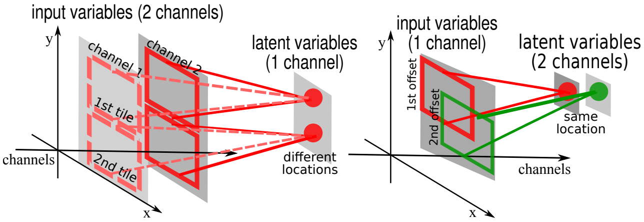

Fig. 7. A toy illustration of how units are combined across layers. Squares are filters, gray planes are channels and circles are latent variables. Left: illustration of how input channels are combined into a single output channel. Input variables at the same spatial location across different channels contribute to determine the state of the same latent variable. Input units falling into different tiles (without overlap) determine the state of nearby units in the hidden layer (here, we have only two spatial locations). Right: illustration of how filters that overlap with an offset contribute to hidden units that are at the same output spatial location but in different hidden channels. In practice, the deep model combines these two methods at each layer to map the input channels into the output channels.

in the image, the representation will be about k times overcomplete[8].

Like other recent papers [47], [48], we propose an intermediate solution that yields a better trade-off between compactness of the parameterization and compactness of the latent representation: *Tiling* [13], [22]. Each local filter is replicated so that it tiles the image without overlaps with itself (*i. e.* it uses a stride that is equal to the diameter of the filter). This reduces the spatial redundancy of the latent variables and allows the input images to have arbitrary size without increasing the number of parameters. To reduce tiling artifacts, different filters typically have different spatial phases so the borders of the different local filters do not all align. In our experiments, we divide the filters into sets with different phases. Filters in the same set are applied to the same image locations (to tile the image), while filters in different sets have a fixed diagonal offset, as shown in fig. 6. In our experiments we only experimented with diagonal offsets but presumably better schemes could be employed, where the whole image is evenly tiled but at the same time the number of overlaps between filter borders is minimized.

At any given layer, the model produces a three-dimensional tensor of values of its latent variables. Two dimensions of this tensor are the dimensions of the 2-D image and the third dimension, which we call a “channel” corresponds to filters with different parameters. The left panel of fig. 7 shows how each unit is connected to all patches centered at its location, across different input channels. The right panel of fig. 7 shows instead that output channels are generated by stacking filter outputs produced by different filters at nearby locations.

When computing the first layer hidden units we stack the precision and the mean hidden units in such a way that units taking input at the same image locations are placed at the same output location but in different channels. Therefore, each second layer RBM hidden unit models cross-correlations among mean and precision units at the first hidden layer.

6 Experiments

In this section we first evaluate the mPoT gated MRF as a generative model by: a) interpreting the filters learned after training on an unconstrained distribution of natural images, b) by drawing samples from the model and c) by using the model on a denoising task[9]. Second, we show that the generative model can be used as a way of learning features by interpreting the expected values of its latent variables as features representing the input. In both cases, generation and feature extraction, the performance of the model is significantly improved by adding layers of binary latent variables on the top of mPoT latent variables. Finally, we show that the generative ability of the model can be used to fill in occluded pixels in images before recognition. This is a difficult task which cannot be handled well without a good generative model of the input data.

6.1 Modeling Natural Images

We generated a dataset of 500,000 color images by picking, at random locations, patches of size 16x16 pixels from images of the Berkeley segmentation dataset[10]. Images were preprocessed by PCA whitening retaining 99% of variance, for a total of 105 projections[11]. We trained a model with 1024 factors, 1024 precision hiddens and 256 mean hiddens using filters of the same size as the input. P was *initialized* with a sparse connectivity inducing a two-dimensional topography when filters in C are laid out on a grid, as shown in fig. 8. Each hidden unit takes as input a neighborhood of filter outputs that learn to extract similar features. Nearby hidden units in the grid use neighborhoods that overlap. Therefore, each precision hidden unit is not only invariant to the sign of the input, because filter outputs are squared, but also it is invariant to a whole subspace of local distortions (that are learned since both C and P are learned). Roughly 90% of the filters learn to be balanced in the R, G, and B channels even though they were randomly initialized. They could be described by localized, orientation-tuned Gabor functions. The colored filters generally have lower spatial frequency and they naturally cluster together in large blobs.

The figure also shows some of the filters representing the mean intensities (columns of matrix W). These features are more complex, with color Gabor filters and on-center off-surround patterns. These features have different effects than the precision filters (see also fig. 4). For instance, a precision filter that resembles an even Gabor is responsible for encouraging interpolation of pixel values along the edge and discouraging interpolation across the edge, while a

[8] In the statistics literature, a convolutional weight-sharing scheme is called a *homogeneous field* while a locally connected one that does not tie the parameters of filters at every location is called *inhomogeneous field*.

[9] Code available at: www.cs.toronto.edu/˜ranzato/publications/mPoT/mPoT.html

[10] Available at: www.cs.berkeley.edu/projects/vision/grouping/segbench/

[11] The linear transform is: $S^{-\frac{1}{2}}U$, where S is a diagonal matrix with eigenvalues on the diagonal entries and U is a matrix whose rows are the leading eigenvectors.

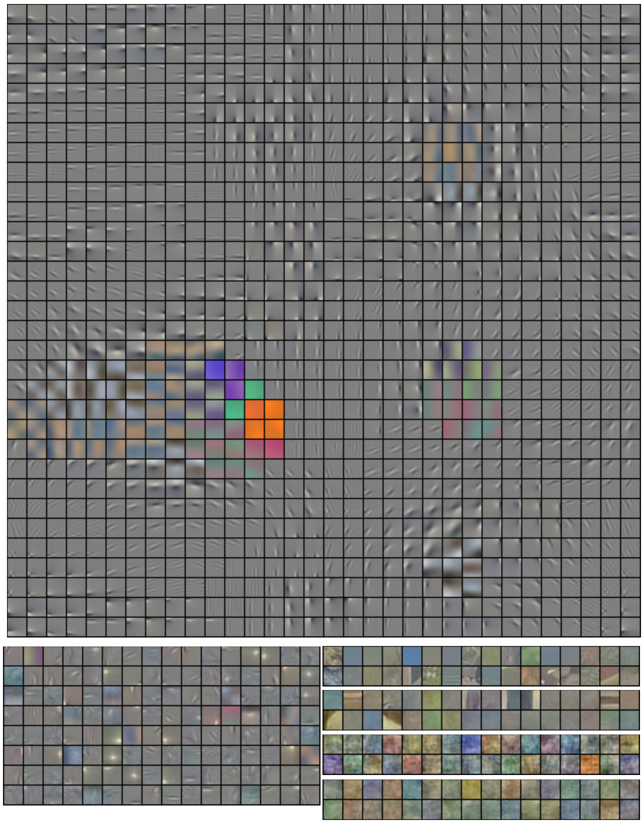

Fig. 8. Top: Precision filters (matrix C) of size 16x16 pixels learned on color image patches of the same size. Matrix P was initialized with a local connectivity inducing a two dimensional topographic map. This makes nearby filters in the map learn similar features. Bottom left: Random subset of mean filters (matrix W). Bottom right (from top to bottom): independent samples drawn from the training data, mPoT, GRBM and PoT.

TABLE 1

Log-probability estimates of test natural images $\mathbf{x}$, and exact log-probability ratios between the same test images $\mathbf{x}$ and random images $\mathbf{n}$.

Model	$\log\left(p(\mathbf{x})\right)$	$\log\left(p(\mathbf{x})/p(\mathbf{n})\right)$
MoG	-85	68
GRBM FPCD	-152	-3
PoT FPCD	-101	65
mPoT CD	-109	82
mPoT PCD	-92	98
mPoT FPCD	-94	102

mean filter resembling a similar Gabor specifies the initial intensity values before the interpolation.

The most intuitive way to test a generative model is to draw samples from it [49]. After training, we then run HMC for a very long time starting from a random image. The bottom right part of Fig. 8 shows that mPoT is actually able to generate samples that look much more like natural images than samples generated by a PoT model, which only models the pair-wise dependencies between pixels or a GRBM model, which only models the mean intensities. The mPoT model succeeds in generating noisy texture patches, smooth patches, and patches with long elongated structures that cover the whole image.

More quantitatively, we have compared the different models in terms of their log-probability on a hold out dataset of test image patches using a technique called annealed importance sampling [50]. Tab. 1 shows great improvements of mPoT over both GRBM and PoT. However, the estimated log-probability is lower than a mixture of Gaussians (MoG) with diagonal covariances and the same number of parameters as the other models. This is consistent with previous work by Theis et al.[51]. We interpret this result with caution since the estimates for GRBM, PoT and mPoT can have a large variance due to the noise introduced by the samplers, while the log-probability of the MoG is calculated exactly. This conjecture is confirmed by the results reported in the rightmost column showing the exact log-probability ratio between the same test images and Gaussian noise with the same covariance[12]. This ratio is indeed larger for mPoT than MoG suggesting inaccuracies in the former estimation. The table also reports results of mPoT models trained by using cheaper but less accurate approximations to the maximum likelihood gradient, namely Contrastive Divergence [52] and Persistent Contrastive Divergence [53]. The model trained with FPCD yields a higher likelihood ratio as expected [44].

We repeated the same data generation experiment using the extension of mPoT to high-resolution images, by training on large patches (of size 238x238 pixels) picked at random locations from a dataset of 16,000 gray-scale natural images that were taken from ImageNet [54][13]. The model was trained using 8x8 filters divided into four sets with different spatial phases. Each set tiles the image with a diagonal offset of two pixels from the previous set. Each set consists of 64 covariance filters and 16 mean filters. Parameters were learned using the algorithm described in sec. 3, but setting P to identity.

The first two rows of fig. 9 compare samples drawn from PoT with samples drawn from mPoT and show that the latter ones exhibit strong structure with smooth regions separated by sharp edges while the former ones lack any sort of long range structure. Yet, mPoT samples still look rather artificial because the structure is fairly primitive and repetitive. We then made the model deeper by adding two layers on the top of mPoT.

All layers are trained by using FPCD but, as training proceeds, the number of Markov chain steps between weight updates is increased from 1 to 100 at the topmost layer in order to obtain a better approximation to the maximum likelihood gradient. The second hidden layer has filters of size 3x3 that also tile the image with a diagonal offset of one. There are 512 filters in each set. Finally, the third layer has filters of size 2x2 and it uses a diagonal offset of one; there are 2048 filters in each set. Every layer performs spatial subsampling by a factor equal to the size of the filters used. This is compensated by an increase in the number of channels which take contributions from the

[12] The log probability ratio is exact since the intractable partition function cancels out. This quantity is equal to the difference of energies.

[13] Categories are: tree, dog, cat, vessel, office furniture, floor lamp, desk, room, building, tower, bridge, fabric, shore, beach and crater.

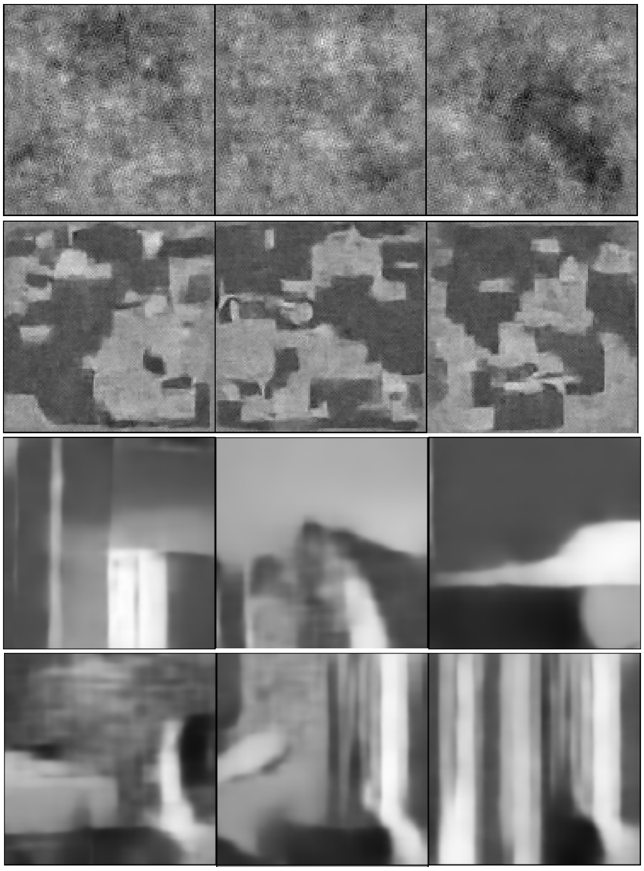

Fig. 9. Top row: Three representative samples generated by PoT after training on an unconstrained distribution of high-resolution gray-scale natural images. Second row: Representative samples generated by mPoT. Third row: Samples generated by a DBN with three hidden layers, whose first hidden layer is the gated MRF used for the second row. Bottom row: as in the third row, but samples (scanned from left to right) are taken at intervals of 50,000 iterations from the same Markov chain. All samples have approximate resolution of 300x300 pixels.

filters that are applied with a different offset, see fig. 7. This implies that at the top hidden layer each latent variable receives input from a very large patch in the input image. With this choice of filter sizes and strides, each unit at the topmost layer affects a patch of size 70x70 pixels. Nearby units in the top layer represent very large (overlapping) spatial neighborhoods in the input image.

After training, we generate from the model by performing 100,000 steps of blocked Gibbs sampling in the topmost RBM (using eq. 15 and 16) and then projecting the samples down to image space as described in sec. 4. Representative samples are shown in the third row of fig. 9. The extra hidden layers do indeed make the samples look more "natural": not only are there smooth regions and fairly sharp boundaries, but also there is a generally increased variety of structures that can cover a large extent of the image. These are among the most realistic samples drawn from models trained on an unconstrained distribution of high-resolution natural images. Finally, the last row of fig. 9 shows the evolution of the sampler. Typically, Markov chains slowly and smoothly evolve morphing one structure into another.

TABLE 2

Denoising performance using $\sigma = 20$ (PSNR=22.1dB).

	Barb.	Boats	Fgpt.	House	Lena	Peprs.
mPoT	28.0	30.0	27.6	32.2	31.9	30.7
mPoT+A	29.2	30.2	28.4	32.4	32.0	30.7
mPoT+A+NLM	30.7	30.4	28.6	32.9	32.4	31.0
FoE [11]	28.3	29.8	-	32.3	31.9	30.6
NLM [55]	30.5	29.8	27.8	32.4	32.0	30.3
GSM [56]	30.3	30.4	28.6	32.4	32.7	30.3
BM3D [57]	31.8	30.9	-	33.8	33.1	31.3
LSSC [58]	31.6	30.9	28.8	34.2	32.9	31.4

6.2 Denoising

The most commonly used task to quantitatively validate a generative model of natural images is image denoising, assuming homogeneous additive Gaussian noise of known variance [10], [11], [37], [58], [12]. We restore images by maximum a-posteriori (MAP) estimation. In the log domain, this amounts to solving the following optimization problem: $\arg\min_{\mathbf{x}} \lambda||\mathbf{y}-\mathbf{x}||^2 + \mathcal{F}(\mathbf{x};\theta)$, where $\mathbf{y}$ is the observed noisy image, $\mathcal{F}(\mathbf{x};\theta)$ is the mPoT energy function (see eq. 12), λ is an hyper-parameter which is inversely proportional to the noise variance and $\mathbf{x}$ is an estimate of the clean image. In our experiments, the optimization is performed by gradient descent.

For images with repetitive texture, generic prior models usually offer only modest denoising performance compared to non-parametric models [45], such as non-local means (NLM) [55] and BM3D [57] which exploit "image self-similarity" in order to denoise. The key idea is to compute a weighted average of all the patches within the test image that are similar to the patch that surrounds the pixel whose value is being estimated, and the current state-of-the-art method for image denoising [58] adapts sparse coding to take that weighted average into account. Here, we adapt the generic prior learned by mPoT in two simple ways: 1) first we adapt the parameters to the denoised test image (mPoT+A) and 2) we add to the denoising loss an extra quadratic term pulling the estimate close to the denoising result of the non-local means algorithm [55] (mPoT+A+NLM). The first approach is inspired by a large body of literature on sparse coding [59], [58] and consists of a) using the parameters learned on a generic dataset of natural images to denoise the test image and b) further adapting the parameters of the model using only the test image denoised at the previous step. The second approach simply consists of adding an additional quadratic term to the function subject to minimization, which becomes $\lambda||\mathbf{y}-\mathbf{x}||^2 + \mathcal{F}(\mathbf{x};\theta) + \gamma||\mathbf{x}_{NLM} - \mathbf{x}||^2$, where $\mathbf{x}_{NLM}$ is the solution of NLM algorithm.

Table 2 summarizes the results of these methods comparing them to the current state-of-the-art methods on widely used benchmark images at an intermediate level of noise. At lower noise levels the difference between these methods becomes negligible while at higher noise levels parametric methods start outperforming non-parametric ones.

First, we observe that adapting the parameters and taking into account image self-similarity improves performance

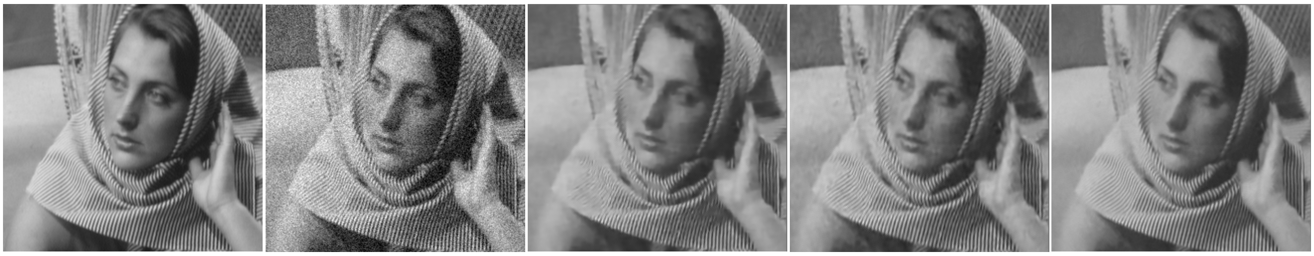

Fig. 10. Denoising of "Barbara" (detail). From left to right: original image, noisy image ($\sigma = 20$, PSNR=22.1dB), denoising of mPoT (28.0dB), denoising of mPoT adapted to this image (29.2dB) and denoising of adapted mPoT combined with non-local means (30.7dB).

(over both baselines, mPoT+A and NLM). Second, on this task a gated MRF with mean latent variables is no better than a model that lacks them, like FoE [11] (which is a convolutional version of PoT)[14]. Mean hiddens are crucial when generating images in an unconstrained manner, but are not needed for denoising since the observation $\mathbf{y}$ already provides (noisy) mean intensities to the model (a similar argument applies to inpainting). Finally, the performance of the adapted model is still slightly worse than the best denoising method.

6.3 Scene Classification

In this experiment, we use a classification task to compare SIFT features with the features learned by adding a second layer of Bernoulli latent variables that model the distribution of latent variables of an mPoT generative model. The task is to classify the natural scenes in the 15 scene dataset [2] into one of 15 categories. The method of reference on this dataset was proposed by Lazebnik et al. [2] and it can be summarized as follows: 1) densely compute SIFT descriptors every 8 pixels on a regular grid, 2) perform K-Means clustering on the SIFT descriptors, 3) compute histograms of cluster ids over regions of the image at different locations and spatial scales, and 4) use an SVM with an intersection kernel for classification.

We use a DBN with an mPoT front-end to mimic this pipeline. We treat the expected value of the latent variables as features that describe the input image. We extract first and second layer features (using the model that produced the generations in the bottom of fig. 9) from a regular grid with a stride equal to 8 pixels. We apply K-Means to learn a dictionary with 1024 prototypes and then assign each feature to its closest prototype. We compute a spatial pyramid with 2 levels for the first layer features ($\{\mathbf{h}^m, \mathbf{h}^p\}$) and a spatial pyramid with 3 levels for the second layer features ($\mathbf{h}^2$). Finally, we concatenate the resulting representations and train an SVM with an intersection kernel for classification. Lazebnik et al. [2] reported an accuracy of 81.4% using SIFT while we obtained an accuracy of 81.2%, which is not significantly different.

6.4 Object Recognition on CIFAR 10

The CIFAR 10 dataset [60] is a hand-labeled subset of a much larger dataset of 80 million tiny images [61], see fig. 11. These images were downloaded from the web and down-sampled to a very low resolution, just 32x32 pixels. The CIFAR 10 subset has ten object categories, namely airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck. The training set has 5000 samples per class, the test set has 1000 samples per class. The low resolution and extreme variability make recognition very difficult and a traditional method based on features extracted at interest-points is unlikely to work well. Moreover, extracting features from such images using carefully engineered descriptors like SIFT [3] or GIST [62] is also likely to be suboptimal since these descriptors were designed to work well on higher resolution images.

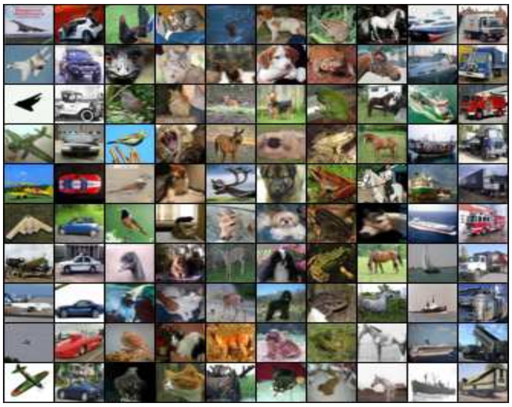

Fig. 11. Example of images in the CIFAR 10 dataset. Each column shows samples belonging to the same category.

TABLE 3
Test and training (in parenthesis) recognition accuracy on the CIFAR 10 dataset. The numbers in italics are the feature dimensionality at each stage.

Method	Accuracy %
1) mean (GRBM): *11025*	59.7 (72.2)
2) cRBM (225 factors): *11025*	63.6 (83.9)
3) cRBM (900 factors): *11025*	64.7 (80.2)
4) mcRBM: *11025*	68.2 (83.1)
5) mcRBM-DBN (*11025-8192*)	70.7 (85.4)
6) mcRBM-DBN (*11025-8192-8192*)	**71.0** (83.6)
7) mcRBM-DBN (*11025-8192-4096-1024-384*)	59.8 (62.0)

We use the following protocol. We train a gated MRF on 8x8 color image patches sampled at random locations, and then we apply the algorithm to extract features convolutionally over the whole 32x32 image by extracting features on a 7x7 regularly spaced grid (stepping every 4 pixels). Then, we use a multinomial logistic regression classifier

[14] The difference of performance between Tiled PoT and convolutional PoT, also called Field of Experts, is not statistically significant on this task.

TABLE 4
Test recognition accuracy on the CIFAR 10 dataset produced by different methods. Features are fed to a multinomial logistic regression classifier for recognition.

Method	Accuracy %
384 dimens. GIST	54.7
10,000 linear random projections	36.0
10K GRBM(*), 1 layer, ZCA'd images	59.6
10K GRBM(*), 1 layer	63.8
10K GRBM(*), 1layer with fine-tuning	64.8
10K GRBM-DBN(*), 2 layers	56.6
11025 mcRBM 1 layer, PCA'd images	**68.2**
8192 mcRBM-DBN, 3 layers, PCA'd images	**71.0**
384 mcRBM-DBN, 5 layers, PCA'd images	**59.8**

to recognize the object category in the image. Since our model is unsupervised, we train it on a set of two million images from the TINY dataset that does not overlap with the labeled CIFAR 10 subset in order to further improve generalization [20], [63], [64]. In the default set up we learn all parameters of the model, we use 81 filters in W to encode the mean, 576 filters in C to encode covariance constraints and we pool these filters into 144 hidden units through matrix P. P is initialized with a two-dimensional topography that takes 3x3 neighborhoods of filters with a stride equal to 2. In total, at each location we extract 144+81=225 features. Therefore, we represent a 32x32 image with a 225x7x7=11025 dimensional descriptor.

Table 3 shows some comparisons. First, we assess whether it is best to model just the mean intensity, or just the covariance or both in 1), 2) and 4). In order to make a fair comparison we used the same feature dimensionality. The covariance part of the model produces features that are more discriminative, but modelling both mean and covariance further improves generalization. In 2) and 3) we show that increasing the number of filters in C while keeping the same feature dimensionality (by pooling more with matrix P) also improves the performance. We can allow for a large number of features as long as we pool later into a more compact and invariant representation. Entries 4), 5) and 6) show that adding an extra layer on the top (by training a binary RBM on the 11025 dimensional feature) improves generalization. Using three stages and 8192 features we achieved the best performance of 71.0%.

We also compared to the more compact 384 dimensional representation produced by GIST and found that our features are more discriminative, as shown in table 4. Previous results using GRBMs [60] reported an accuracy of 59.6% using whitened data while we achieve 68.2%. Their result improved to 64.8% by using unprocessed data and by fine-tuning, but it did not improve by using a deeper model. Our performance improves by adding other layers showing that these features are more suitable for use in a DBN. The current state-of-the-art result on this dataset is 80.5% [65] and it employs a much bigger network and perturbation of the input to improve generalization.

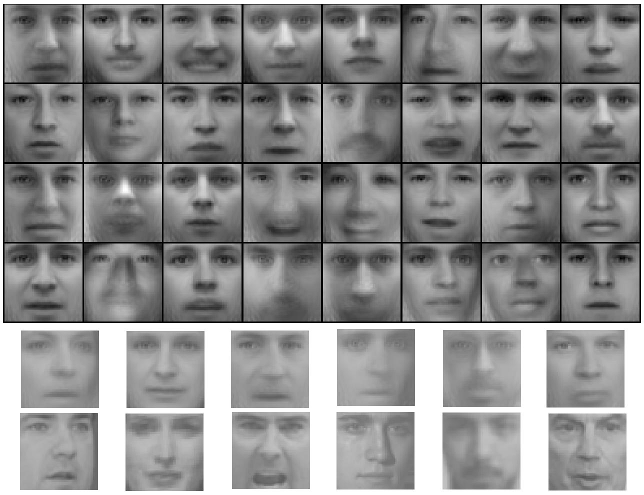

Fig. 12. Top: Samples generated by a five-layer deep model trained on faces. The top layer has 128 binary latent variables and images have size 48x48 pixels. Bottom: comparison between six samples from the model (top row) and the Euclidean distance nearest neighbor images in the training set (bottom row).

TABLE 5
TFD: Facial expression classification accuracy using features trained without supervision.

Method	layer 1	layer 2	layer 3	layer 4
raw pixels	71.5	-	-	-
Gaussian SVM	76.2	-	-	-
Sparse Coding	74.6	-	-	-
Gabor PCA	80.2	-	-	-
GRBM	80.0	81.5	80.3	79.5
PoT	79.4	79.3	80.6	80.2
mPoT	81.6	82.1	**82.5**	82.4

6.5 Recognition of Facial Expressions

In these experiments we study the recognition of facial expressions under occlusion in the Toronto Face Database (TFD) [66]. This is the largest publicly available dataset of faces to date, created by merging together 30 pre-existing datasets [66]. It has about 100,000 images that are unlabeled and more than 4,000 images that are labeled with seven facial expressions, namely: anger, disgust, fear, happiness, sadness, surprise and neutral. Faces were preprocessed by: detection and alignment of faces using the Machine Perception Toolbox [67], followed by down-sampling to a common resolution of 48x48 pixels. We choose to predict facial expressions under occlusion because this is a particularly difficult problem: The expression is a subtle property of faces that requires good representations of detailed local features, which are easily disrupted by occlusion.

Since the input images have fairly low resolution and the statistics across the images are strongly non-stationary (because the faces have been aligned), we trained a deep model *without* weight-sharing. The first layer uses filters of size 16x16 centered at grid-points that are four pixels apart, with 32 covariance filters and 16 mean filters at each grid-point. At the second layer we learn a fully-connected RBM with 4096 latent variables each of which

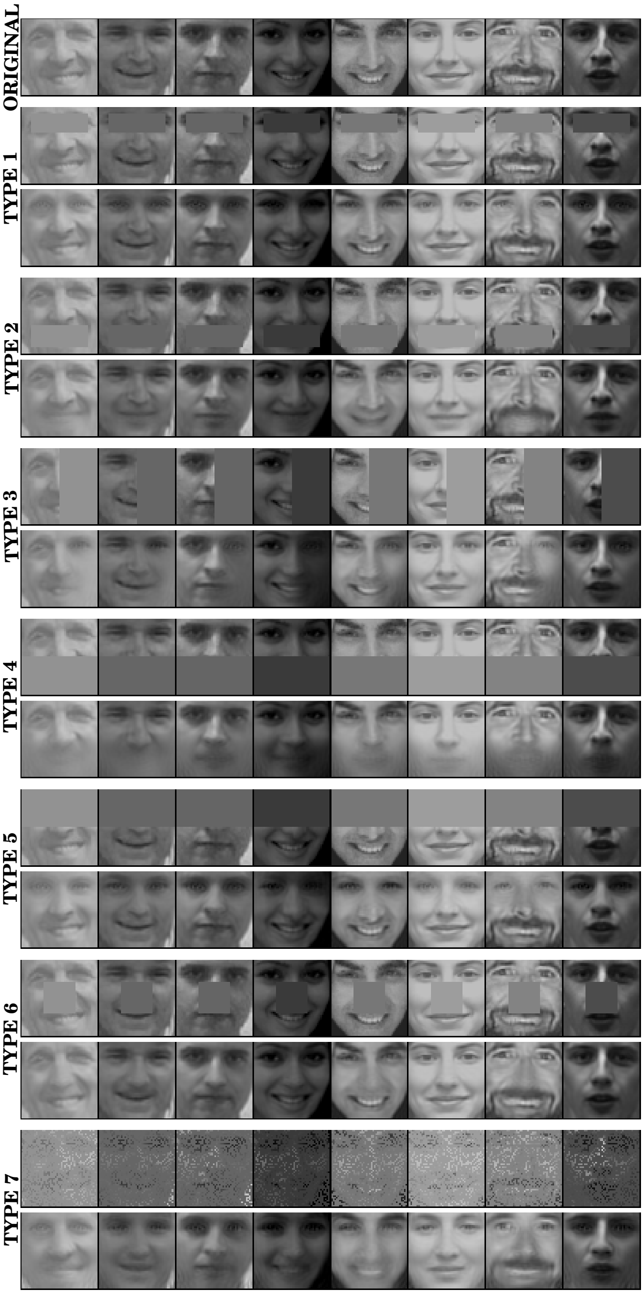

Fig. 13. Example of conditional generation performed by a four-layer deep model trained on faces. Each column is a different example (not used in the unsupervised training phase). The topmost row shows some example images from the TFD dataset. The other rows show the same images occluded by a synthetic mask (on the top) and their restoration performed by the deep generative model (on the bottom).

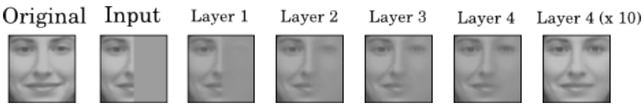

Fig. 14. An example of restoration of unseen images performed by propagating the input up to first, second, third, fourth layer, and again through the four layers and re-circulating the input through the same model for ten times.

is connected to all of the first layer features. Similarly, at the third and fourth layer we learn fully connected

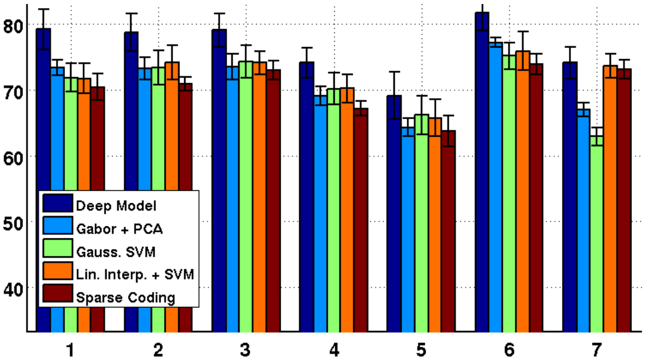

Fig. 15. Expression recognition accuracy on the TFD dataset when both training and test labeled images are subject to 7 types of occlusion.

RBMs with 1024 latent variables, and at the fifth layer we have an RBM with 128 hidden units. The deep model was trained entirely generatively on the *unlabeled* images without using any labeled instances [64]. The discriminative training consisted of training a linear multi-class logistic regression classifier on the top level representation *without* using back-propagation to jointly optimize the parameters across all layers.

Fig. 12 shows samples randomly drawn from the generative model. Most samples resemble plausible faces of different individuals with a nice variety of facial expressions, poses, and lighting conditions. Nearest neighbor analysis, using Euclidean distance, reveals that these samples are not just copies of images in the training set: Each sample exhibits regularities derived from many different training cases.

In the first experiment, we train a linear classifier on the features produced by each layer and we predict the facial expression of the images in the labeled set. Each input image is processed by subtracting from each pixel the mean intensity in that image then dividing by the standard deviation of the pixels in that image. The features at successive hidden layers give accuracies of 81.6%, 82.1%, 82.5%, and 82.4%. Each higher layer is a RBM with 4096 hidden units. These accuracies should be compared to: 71.5% achieved by a linear classifier on the raw pixels, 76.2% achieved by a Gaussian SVM on the raw pixels, 74.6% achieved by a sparse coding method [68], and 80.2% achieved by the method proposed by Dailey et al. [69] which employs a large Gabor filter bank followed by PCA and a linear classifier, using cross-validation to select the number of principal components. The latter method and its variants [70] are considered a strong baseline in the literature. Table 5 reports also a direct comparison to a DBN with a GRBM at the first layer and to a DBN with a PoT at the first layer using the same number of parameters and weight sharing scheme. mPoT outperforms its competitors on this task. The accuracies reported on the table are an average over 5 random training/test splits of the data with 80% of the images used for training and the rest for test. Facial identities of subjects in training and test sets are disjoint.

In the next experiment, we apply synthetic occlusions only to the labeled images. The occlusions are shown in fig. 13, they block: 1) eyes, 2) mouth, 3) right half, 4) bottom half, 5) top half, 6) nose and 7) 70% of the pixels at random. Before extracting the features, we use the generative model to *fill-in* the missing pixels, assuming knowledge of which pixels are occluded. In order to fill-in we initialize the missing pixels at zero and propagate the occluded image through the four layers using the sequence of posterior expectations. Then we reconstruct from the top layer representation using the sequence of conditional expectations in the generative direction. The last step of the reconstruction consists of using the mPoT model to fill in only the missing pixels by conditioning on both the known pixels and the first-layer hidden variables which gives a Gaussian distribution for the missing pixels [12]. This whole up and down process is repeated a few times with the number of times being determined by the filling-in performance (in terms of PSNR) on a validation set of unlabeled images.

Fig. 14 shows the filling process. The latent representation in the higher layers is able to capture longer range structure and it does a better job at filling-in the missing pixels[15]. After missing pixels are imputed, the model is used to extract features from the restored images as before.

The results reported in fig. 15 show that the deep model is generally more robust to these occlusions, even compared to other methods that know which pixels are missing and try to compensate for occlusion. In these figures, we compare to a Gaussian SVM on the raw images, a Gaussian SVM on linearly interpolated images, a Gabor-based approach [69] on linearly interpolated images and a sparse coding approach on the unoccluded part of the input images [68].

7 Summary and Future Work

Gated MRFs are higher-order MRFs that can be more easily described as MRFs with latent variables. The joint distribution is a product of potentials that either take pairs of variables (one input and one "mean" latent variable) or triplets of variables (two inputs and one "precision" latent variable). For every configuration of latent variables (and there are exponentially many if these are binary, for instance) the model describes the input with a multi-variate Gaussian distribution with a certain mean and covariance matrix. The conditional distribution over the input induces a soft-partitioning of the input space (one for each configuration of latent variables) with hyper-ellipses that can fit much more precisely the input distribution than using hyper-spheres (like in PPCA [25] or GRBM [29]) or hyper-ellipses centered at the origin (like in PoT [30]).

We have described a probabilistic generative model that can be used for a large variety of applications in both low-level and high-level vision. In both cases, better performance is achieved by making the model hierarchical which is easily done by using the latent variables of the gated MRF as the first layer of a DBN that uses Bernoulli variables for all subsequent layers. The ability of the model to generate realistic samples can be used to assess the quality of learning and to intuitively see what information is captured or lost during training.

For high-level vision tasks such as object or scene recognition, the latent variables of the model can be used as image descriptors. These descriptors offer several advantages over engineered descriptors. They can *adapt* to the input domain by leveraging large amounts of unlabeled data (as done in this work) but they can also be tuned to the task at hand by back-propagating the discriminative error through the feature extractor. Feature extraction in the hierarchical model is computationally efficient and is equivalent to feed-forward propagation in a neural network with a peculiar first layer composed of mean units that act as linear filters followed by a logistic non-linearity and precision units that pool the squared outputs of many linear filters. This contrasts with inference in directed graphical models which typically requires iteration, or inaccurate variational approximations[16]. Generating unbiased samples from a directed model is typically much simpler than generating from our model, but for computer vision, efficient inference is far more important than efficient generation.

The main drawbacks of our model are that exact maximum likelihood learning is intractable and so computationally expensive Markov Chain Monte Carlo methods have to be used during training. This increases the number of hyper-parameters that have to be set and makes the training slow and hard to monitor. Consequently, the assessment of whether our model is preferable to others seems to be application-dependent since we must consider the trade-off between the computational cost of training and the efficiency of inference at test time.

Although our model generates quite good samples of natural images, these samples still exhibit rather simplistic structure and are very limited in the types of texture they contain. The gated MRF can be extended in several ways to address these issues by a) modifying the form of the energy function to better embed our prior knowledge of natural image statistics and to better model texture and long range dependencies (e.g., learning multi-scale representations), b) improving the model at the higher layers of the hierarchy (e.g., replacing RBMs with gated RBMs that are the analogue of the model here described but for binary input variables) and c) exploiting image self-similarity by designing mixed parametric and non-parametric models to more naturally represent repetitive texture. Finally, a very promising research avenue is to extend the model to video sequences in which the temporal regularities created by smoothly changing viewing transformations should make it far easier to learn to model depth, three-dimensional transformations and occlusion [71].

[15] A similar experiment using a similar DBN was also reported by Lee et al. [17] in fig. 6 of their paper. Our generation is more realistic probably thanks to the better modeling of the first layer mPoT.

[16] Feedforward inference in our hierarchical generative model can be viewed as a type of variational approximation that is only exactly correct for the top layer, but the inference for the lower layers is a very good approximation because of the way they are learned [20].

ACKNOWLEDGMENTS

The research was funded by grants from NSERC, CFI and CIFAR and by gifts from Google and Microsoft.

REFERENCES

[1] E. Simoncelli, "Statistical modeling of photographic images," *Handbook of Image and Video Processing*, pp. 431–441, 2005.

[2] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2006.

[3] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision and Image Understanding*, 2008.

[6] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *CIVR*, 2007.

[7] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. John Wiley & Sons, 2001.

[8] G. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] M. Ranzato and G. Hinton, "Modeling pixel means and covariances using factorized third-order boltzmann machines," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[10] M. Wainwright and E. Simoncelli, "Scale mixtures of gaussians and the statistics of natural images," in *Advances in Neural Information Processing Systems*, 2000.

[11] S. Roth and M. Black, "Fields of experts: A framework for learning image priors," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[12] U. Schmidt, Q. Gao, and S. Roth, "A generative perspective on mrfs in low-level vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[13] M. Ranzato, V. Mnih, and G. Hinton, "Generating more realistic images using gated mrf's," in *Advances in Neural Information Processing Systems*, 2010.

[14] Y. Karklin and M. Lewicki, "Emergence of complex cell properties by learning to generalize in natural scenes," *Nature*, vol. 457, pp. 83–86, 2009.

[15] U. Koster and A. Hyvarinen, "A two-layer ica-like model estimated by score matching," in *ICANN*, 2007.

[16] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *International Conference in Machine Learning*, 2008.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng., "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. International Conference in Machine Learning*, 2009.

[18] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *International Conference in Computer Vision*, 2009.

[19] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[20] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[21] M. Ranzato, A. Krizhevsky, and G. Hinton, "Factored 3-way restricted boltzmann machines for modeling natural images," in *Conference in Artificial Intelligence and Statistics*, 2010.

[22] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton, "On deep generative models with applications to recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[23] J. DiCarlo, D. Zoccolan, and R. N.C., "How does the brain solve visual object recognition?" *Neuron*, vol. 73, no. 3, pp. 415–34, 2012.

[24] M. Ranzato, "Unsupervised learning of feature hierarchies," Ph.D. thesis, ch. 1, 2009.

[25] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, Series B*, vol. 61, pp. 611–622, 1999.

[26] G. Young, "Maximum likelihood estimation and factor analysis," *Psychometrika*, vol. 6, no. 1, pp. 49–53, 1940.

[27] D. MacKay, "Maximum likelihood and covariant algorithms for independent component analysis," 1999.

[28] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by v1?" *Vision Research*, vol. 37, pp. 3311–3325, 1997.

[29] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Advances in Neural Information Processing Systems*, 2005.

[30] M. Welling, G. Hinton, and S. Osindero, "Learning sparse topographic representations with products of student-t distributions," in *Advances in Neural Information Processing Systems*, 2003.

[31] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton, "Energy-based models for sparse overcomplete representations," *JMLR*, vol. 4, pp. 1235–1260, 2003.

[32] T. Sejnowski, "Higher-order boltzmann machines," in *AIP Conf. proc., Neural networks for computing*, 1986.

[33] R. Memisevic and G. Hinton, "Learning to represent spatial transformations with factored higher-order boltzmann machines." *Neural Computation*, vol. 22, pp. 1473–1492, 2009.

[34] M. Welling and G. E. Hinton, "A new learning algorithm for mean field boltzmann aachines," in *Int. Conf. Artificial Neural Networks*, 2002.

[35] S. Osindero and G. E. Hinton, "Modeling image patches with a directed hierarchy of markov random fields," in *Advances in Neural Information Processing Systems*, 2008.

[36] G. Taylor, G. Hinton, and S. Roweis, "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems*, 2007.

[37] Y. Weiss and W. Freeman, "What makes a good model of natural images?" in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[38] C. Williams and F. Agakov, "Products of gaussians and probabilistic minor component analysis," *Neural Computation*, vol. 14, pp. 1169–1182, 2002.

[39] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *PAMI*, vol. 6, pp. 721–741, 1984.

[40] M. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *Int. Journal of Computer Vision*, vol. 19, no. 1, pp. 57–92, 1996.

[41] G. Hinton and Y. Teh, "Discovering multiple constraints that are frequently approximately satisfied," in *Uncertainty and Artificial Intelligence*, 2001.

[42] G. Hinton, "Products of experts," in *Proc. of the Ninth International Conference on Artificial Neural Networks*, 1999.

[43] R. Neal, *Bayesian learning for neural networks*. Springer-Verlag, 1996.

[44] T. Tieleman and G. Hinton, "Using fast weights to improve persistent contrastive divergence," in *International Conference in Machine Learning*, 2009.

[45] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[47] K. Gregor and Y. LeCun, "Emergence of complex-like cells in a temporal product network with local receptive fields," arXiv:1006.0448, 2010.

[48] Q. Le, J. Ngiam, Z. Chen, D. Chia, P. Koh, and A. Ng, "Tiled convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2010.

[49] S. Zhu and D. Mumford, "Prior learning and gibbs reaction diffusion," *PAMI*, pp. 1236–1250, 1997.

[50] I. Murray and R. Salakhutdinov, "Evaluating probabilities under high-dimensional latent variable models," 2009.

[51] L. Theis, S. Gerwinn, F. Sinz, and M. Bethge, "In all likelihood, deep belief is not enough," *Journal of Machine Learning Research*, vol. 12, pp. 3071–3096, 2011.

[52] M. A. Carreira-Perpignan and G. E. Hinton, "On contrastive divergence learning," *Artificial Intelligence and Statistics*, 2005.

[53] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *International Conference in Machine Learning*, 2008.

[54] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[55] A. Buades, B. Coll, and J. Morel, "A non local algorithm for image denoising," in *IEEE Computer Vision and Pattern Recognition*, 2005.

[56] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Trans. Image Processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[57] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3d filtering," in *Proc. SPIE Electronic Imaging*, 2006.

[58] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Int. Conference on Computer Vision*, 2009.

[59] M. Elad and M. Aharon, "Image denoising via learned dictionaries and sparse representation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[60] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009, mSc Thesis, Dept. of Comp. Science, Univ. of Toronto.

[61] A. Torralba, R. Fergus, and W. Freeman, "80 million tiny images: a large dataset for non-parametric object and scene recognition," *PAMI*, vol. 30, pp. 1958–1970, 2008.

[62] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *IJCV*, vol. 42, pp. 145–175, 2001.

[63] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[64] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *International Conference in Machine Learning*, 2007.

[65] D. Ciresan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI*, 2011.

[66] J. M. Susskind, A. K. Anderson, and G. E. Hinton, "The Toronto face database," Department of Computer Science, University of Toronto, Toronto, ON, Canada, Tech. Rep., 2010.

[67] B. Fasel, I. Fortenberry and J. Movellan, "A generative framework for real-time object detection and classification," in *CV Image Understanding*, 2005.

[68] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI*, 2008.

[69] M. Dailey, G. Cottrell, R. Adolphs, and C. Padgett, "Empath: A neural network that categorizes facial expressions," *Journal of Cognitive Neuroscience*, vol. 14, pp. 1158–1173, 2002.

[70] G. Littlewort, M. Bartlett, I. Fasel, J. Susskind, and J. Movellan, "Dynamics of facial expression extracted automatically from video," *Computer Vision and Pattern Recognition Workshop*, vol. 5, p. 80, 2004.

[71] G. Hinton, A. Krizhevsky, and S. Wang, "Learning structural descriptions of objects using equivariant capsules," in *Int. Conf. on Artificial Neural Networks*, 2011.

Marc'Aurelio Ranzato is a research scientist at Google. He received a PhD in computer science in 2009 under the supervision of professor Yann LeCun at New York University and he later was a postdoctoral fellow working with professor Geoffrey Hinton in the department of computer science at the University of Toronto. He is recipient of the 2008-2009 NYU Dean's dissertation fellowship. Marc'Aurelio's major interests are in the areas of machine learning, computer vision and multi-media processing.

Volodymyr Mnih received a BSc in mathematics from the University of Toronto and an MSc in computing science from the University of Alberta. He is currently a PhD student in computer science under the supervision of Geoffrey Hinton at the University of Toronto. His main research interests are in machine learning and computer vision.

Joshua M. Susskind is a postdoctoral scholar working with professors Marian Bartlett and Javier Movellan in the Institute for Neural Computation at the University of California, San Diego. He received a PhD in Psychology in 2011 under the co-supervision of professors Adam Anderson (Cognitive Neuroscience) and Geoffrey Hinton (Computer Science) at the University of Toronto. His research interests are multi-disciplinary, using experimental psychology and machine learning to understand how the brain perceives facial expressions.

Geoffrey E. Hinton received a BA in Experimental Psychology from Cambridge in 1970 and a PhD in Artificial Intelligence from Edinburgh in 1978. He was a member of the PDP group at the University of California, San Diego and an assistant and associate professor at Carnegie-Mellon University. He is currently the Raymond Reiter Distinguished Professor of Artificial Intelligence at the University of Toronto and the director of the program on Neural Computation and Adaptive Perception funded by the Canadian Institute for Advanced Research. From 1998 to 2001 he set up the Gatsby Computational Neuroscience Unit at University College, London. He is a fellow of the Royal Society and the Royal Society of Canada, an honorary foreign member of the American Academy of Arts and Sciences, and a former president of the Cognitive Science Society. He has received the the David E. Rumelhart prize, the IJCAI research excellence award, the Gerhard Herzberg Canada gold medal for Science and Engineering, and honorary doctorates from the Universities of Edinburgh and Sussex. His research contributions include backpropagation, Boltzmann machines, distributed representations, time-delay neural nets, mixtures of experts, variational inference and learning, contrastive divergence, and deep belief nets.