The Hierarchical Beta Process for Convolutional Factor Analysis and Deep Learning

Bo Chen[1] BC69@DUKE.EDU
Gungor Polatkan[2] POLATKAN@PRINCETON.EDU
Guillermo Sapiro[3] GUILLE@ECE.UMN.EDU
David B. Dunson[4] DUNSON@STAT.DUKE.EDU
Lawrence Carin[1] LCARIN@EE.DUKE.EDU

1. Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
2. Departments of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA
3. Department of Electrical and Computer Engineering, University of Minnesota, MN 55455, USA
4. Department of Statistical Science, Duke University, Durham, NC 27708, USA

Abstract

A convolutional factor-analysis model is developed, with the number of filters (factors) inferred via the beta process (BP) and hierarchical BP, for single-task and multi-task learning, respectively. The computation of the model parameters is implemented within a Bayesian setting, employing Gibbs sampling; we explicitly exploit the convolutional nature of the expansion to accelerate computations. The model is used in a multi-level ("deep") analysis of general data, with specific results presented for image-processing data sets, e.g., classification.

1. Introduction

There has been significant recent interest in multi-layered or "deep" models for representation of general data, with a particular focus on imagery and audio signals. These models are typically implemented in a hierarchical manner, by first learning a data representation at one scale, and using the model weights or parameters learned at that scale as inputs for the next level in the hierarchy. Methods that have been considered include deconvolutional networks (Zeiler et al., 2010), convolutional networks (LeCun et al.), deep belief networks (DBNs) (Hinton et al.), hierarchies of sparse auto-encoders (Jarrett et al., 2009; Ranzato et al., 2006; Vincent et al., 2008; Erhan et al., 2010), and convolutional restricted Boltzmann machines (RBMs) (Lee et al., 2009a;b; Norouzi et al., 2009). A key aspect of many of these algorithms is the exploitation of the convolution operator, which plays an important role in addressing large-scale problems, as one must typically consider all possible shifts of canonical filters. In such analysis one must learn the form of the filter, as well as the associated coefficients. Concerning the latter, it has been recognized that a preference for sparse coefficients is desirable (Zeiler et al., 2010; Lee et al., 2009b; Norouzi et al., 2009; Lee et al., 2008).

Some of the multi-layered models have close connections to over-complete dictionary learning (Mairal et al., 2009), in which image patches are expanded in terms of a sparse set of dictionary elements. The deconvolutional and convolutional networks in (Zeiler et al., 2010; Lee et al., 2009a; Norouzi et al., 2009) similarly represent each level of the hierarchical model in terms of a sparse set of dictionary elements; however, rather than separately considering distinct patches as in (Mairal et al., 2009), the work in (Zeiler et al., 2010; Lee et al., 2009a) allows all possible shifts of dictionary elements for representation of the entire image at once (not separate patches). In the context of over-complete dictionary learning, researchers have also considered multi-layered or hierarchical models, but again in terms of image patches (Jenatton et al., 2010).

All of the methods discussed above, for "deep" models and for sparse dictionary learning for image patches, require one to specify *a priori* the number of filters or dictionary elements employed within each layer of the model. In many applications it may be desirable to infer the number of filters based on the data itself. This

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

corresponds to a problem of inferring the proper number of features for the data of interest, while allowing for all possible shifts of the filters, as in the various convolutional models discussed above. The idea of learning an appropriate number and composition of features has motivated the Indian buffet process (IBP) (Griffiths & Ghahramani, 2005), as well as the beta process (BP) to which it is closely connected (Thibaux & Jordan, 2007; Paisley & Carin, 2009). Such methods have been applied recently to (single-layer) dictionary learning in the context of image patches (Zhou et al., 2009). Further, the IBP has recently been employed for design of "deep" graphical models (Adams et al., 2010), although the problem considered in (Adams et al., 2010) is distinct from that associated with the deep models discussed above.

In this paper we demonstrate that the idea of building an unsupervised deep model may be cast in terms of a hierarchy of *convolutional* factor-analysis models, with the factor scores from layer l serving as the input to layer $l+1$. The framework presented here has four key differences with previous deep unsupervised models: (i) the number of filters at each layer of the deep model is inferred from the data by an IBP/BP construction; (ii) multi-task feature learning is performed for simultaneous analysis of different families of images, using the *hierarchical* beta process (HBP) (Thibaux & Jordan, 2007); (iii) fast computations are performed using Gibbs sampling, where the convolution operation is exploited directly within the update equations; and (iv) sparseness is imposed on the filter coefficients and filters themselves, via a Bayesian generalization of the ℓ_1 regularizer. In the experimental section, we also give a detailed analysis on the role of sparseness on different parameters in our deep model.

2. Convolutional Sparse Factor Analysis

2.1. Single task learning & the beta process

We consider N images $\{\mathbf{X}_n\}_{n=1,N}$, and all images are analyzed jointly; the images are assumed drawn from the same (single) statistical model, with this termed "single-task" learning. The nth image to be analyzed is $\mathbf{X}_n \in \mathbb{R}^{n_y \times n_x \times K_c}$, where K_c is the number of color channels (*e.g.*, for gray-scale images $K_c = 1$, while for RGB images $K_c = 3$). Each image is expanded in terms of a dictionary, with the dictionary defined by compact canonical elements $\boldsymbol{d}_k \in \mathbb{R}^{n'_y \times n'_x \times K_c}$, with $n'_x \ll n_x$ and $n'_y \ll n_y$; the shifted $\boldsymbol{d}_k$ corresponds to the kth filter. The dictionary elements are designed to capture local structure within $\mathbf{X}_n$, and all possible two-dimensional (spatial) shifts of the dictionary elements are considered for representation of $\mathbf{X}_n$. The shifted dictionary elements are assumed zero-padded spatially, such that they are matched to the size of $\mathbf{X}_n$. For K canonical dictionary elements the cumulative dictionary is $\mathcal{D} = \{\boldsymbol{d}_k\}_{k=1,K}$. In practice the number of dictionary elements K is made large, and we wish to infer the subset of $\mathcal{D}$ that is actually needed to sparsely render $\mathbf{X}_n$ as

$$\mathbf{X}_n = \sum_{k=1}^{K} b_{nk} \mathbf{W}_{nk} * \boldsymbol{d}_k + \boldsymbol{\epsilon}_n \tag{1}$$

where $*$ is the convolution operator and $b_{nk} \in \{0,1\}$ indicates whether $\boldsymbol{d}_k$ is used to represent $\mathbf{X}_n$, and $\boldsymbol{\epsilon}_n \in \mathbb{R}^{n_y \times n_x \times K_c}$ represents the residual; The $\mathbf{W}_{nk}$ represents the weights of dictionary k for image $\mathbf{X}_n$; the support of $\mathbf{W}_{nk}$ is $(n_y - n'_y + 1) \times (n_x - n'_x + 1)$, allowing for all possible shifts, as in a typical convolutional model (Lee et al., 2009a). We impose within the model that the $\{w_{nki}\}_{i \in \mathcal{S}}$ are sparse or nearly sparse, such that most w_{nki} are sufficiently small to be discarded without significantly affecting the reconstruction of $\mathbf{X}_n$, where the set $\mathcal{S}$ contains all possible indexes for dictionary shifts. A similar sparseness constraint was imposed in (Zeiler et al., 2010; Norouzi et al., 2009; Lee et al., 2009b; 2008). The model can be implemented via the following hierarchical construction:

$$\begin{aligned} b_{nk} &\sim \text{Bernoulli}(\pi_k)\ , \quad \pi_k \sim \text{Beta}(1/K, b) \\ w_{nki} &\sim \mathcal{N}(0, 1/\alpha_{nki})\ , \quad \alpha_{nki} \sim \text{Gamma}(e, f) \\ \boldsymbol{d}_k &\sim \prod_{j=1}^{J} \mathcal{N}(0, 1/\beta_j), \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_P \gamma_n^{-1}) \end{aligned} \tag{2}$$

where J denotes the number of pixels in the dictionary element $\boldsymbol{d}_k$ and d_{kj} is the jth component of $\boldsymbol{d}_k$; hyperpriors $\gamma_n \sim \text{Gamma}(c, d)$ and $\beta_j \sim \text{Gamma}(g, h)$ are also employed. The integer P denotes the number of pixels in $\mathbf{X}_n$, and $\mathbf{I}_P$ represents a $P \times P$ identity matrix. The hyperparameters (e, f) and (g, h) are set to favor large α_{nki} and β_j, thereby imposing that the set of w_{nki} will be compressible or approximately sparse, with the same found useful for the dictionary elements $\boldsymbol{d}_k$. In the limit $K \to \infty$, and upon marginalizing out $\{\pi_k\}_{k=1,K}$, the above model corresponds to the Indian buffet process (IBP) (Griffiths & Ghahramani, 2005; Thibaux & Jordan, 2007).

Following notation from (Thibaux & Jordan, 2007), we write that for image n we draw $X_n \sim \text{BeP}(B)$, with $B \sim \text{BP}(b, B_0)$, where the base probability measure is $B_0 = \prod_{j=1}^{J} \mathcal{N}(0, 1/\beta_j)$, $\text{BP}(\cdot)$ represents the beta process, and $\text{BeP}(\cdot)$ represents the Bernoulli process. Assuming $K \to \infty$, the $X_n = \sum_{k=1}^{\infty} b_{nk} \delta_{\boldsymbol{d}_k}$ defines which of the dictionary elements $\boldsymbol{d}_k$ are employed to represent image n, and $B = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{d}_k}$; $\delta_{\boldsymbol{d}_k}$ is a unit point measure concentrated at $\boldsymbol{d}_k$. The $\{\pi_k\}$ are drawn

from a *degenerate* beta distribution with parameter b (Thibaux & Jordan, 2007). In this single-task construction all images have the same probability π_k of employing filter $\boldsymbol{d}_k$.

2.2. Multitask learning & the hierarchical BP

Assume we have T learning tasks, where task $t \in \{1, \ldots, T\}$ is defined by the set of images $\{\mathbf{X}_n^{(t)}\}_{n=1,N_t}$, where N_t is the number of images in task t. The T "tasks" may correspond to distinct but related types of images. We wish to learn a model of the form

$$\mathbf{X}_n^{(t)} = \sum_{k=1}^{K} b_{nk}^{(t)} \mathbf{W}_{nk}^{(t)} * \boldsymbol{d}_k + \boldsymbol{\epsilon}_n^{(t)} \tag{3}$$

Note that the dictionary elements $\{\boldsymbol{d}_k\}$ are shared across all tasks, and the task-dependent $b_{nk}^{(t)}$ defines whether $\boldsymbol{d}_k$ is used in image n of task t. The $X_n^{(t)} = \sum_{k=1}^{\infty} b_{nk}^{(t)} \delta_{\boldsymbol{d}_k}$, defining filter usage for image n in task t, are constituted via an HBP construction:

$$X_n^{(t)} \sim \text{BeP}(B^{(t)}) \tag{4}$$

$$B^{(t)} \sim \text{BP}(b_2, B) \ , \quad B \sim \text{BP}(b_1, B_0) \tag{5}$$

with B_0 defined as above. Note that via this construction each $B^{(t)} = \sum_{k=1}^{\infty} \pi_k^{(t)} \delta_{\boldsymbol{d}_k}$ shares the same filters, but with task-specific probability of filter usage, $\{\pi_k^{(t)}\}$. Therefore, this model imposes that the different tasks may share usage of filters $\{\boldsymbol{d}_k\}$, but the priority with which filters are used varies across tasks. The measure $B = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{d}_k}$ defines the probability with which filters are used across all images and tasks, with $\boldsymbol{d}_k$ employed with probability π_k.

When presenting results below, we refer to $B = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{d}_k}$ as constituting the "global" buffet of atoms and associated probabilities, across all tasks. The measure $B^{(t)}$ is a "local" representation specifically for task t (with the same atoms as B, but with task-dependent probability of atom usage).

2.3. Exploiting convolution in computations

In this paper we present results based on Gibbs sampling; however, we have also implemented variational Bayesian (VB) analysis and achieved very similar results. In both cases all update equations are analytic, as a result of the conjugate-exponential nature of consecutive equations in the hierarchical model. Below we focus on Gibbs inference, for the special case of single-task learning (for simplicity of presentation), and discuss the specific update equations in which convolution is leveraged; related equations hold for the HBP model, and for VB inference.

To sample b_{nk} and $\mathbf{W}_{nk}$ (*i.e.*, $\{w_{nki}\}_{i \in \mathcal{S}}$), we have $p(b_{nk} = 1|-) = \widetilde{\pi}_{nk}$, and $p(w_{nki}, i \in \mathcal{S}|-) = (1 - b_{nk})\mathcal{N}(0, \alpha_{nki}^{-1}) + b_{nk}\mathcal{N}(\mu_{nki}, \Sigma_{nki})$, where $\Sigma_{nki} = (\boldsymbol{d}_{ki}^T \boldsymbol{d}_{ki} \gamma_n + \alpha_{nki})^{-1}$, $\mu_{nki} = \Sigma_{nki} \gamma_n \mathbf{X}_{nki}^T \boldsymbol{d}_{ki}$, with $\mathbf{X}_{nki} = \mathbf{X}_{-n} + b_{nk} \boldsymbol{d}_{ki} w_{nki}$, and

$$\frac{\widetilde{\pi}_{nk}}{1 - \widetilde{\pi}_{nk}} = \frac{\pi_k}{1 - \pi_k} \cdot \frac{\mathcal{N}(\mathbf{X}_{nk}|\mathbf{W}_{nk} * \boldsymbol{d}_k, \gamma_n^{-1}\mathbf{I}_P)}{\mathcal{N}(\mathbf{X}_{nk}|\mathbf{0}, \gamma_n^{-1}\mathbf{I}_P)}$$

Here $\mathbf{X}_{nk} = \mathbf{X}_{-n} + \mathbf{W}_{nk} * \boldsymbol{d}_k$, $\mathbf{X}_{-n} = \mathbf{X}_n - \sum_{k=1}^{K} b_{nk} \mathbf{W} * \boldsymbol{d}_k$ and b_{nk} is the most recent sample.

Taking advantage of the convolution property, we simultaneously update the posterior mean and covariance of the coefficients for all the shifted versions of one dictionary element. Consequently,

$$\begin{aligned} \boldsymbol{\Sigma}_{nk} &= \mathbf{1} \oslash (\gamma_n \|\boldsymbol{d}_k\|_2^2 + \boldsymbol{\alpha}_{nk}) \\ \boldsymbol{\mu}_{nk} &= \gamma_n \boldsymbol{\Sigma}_{nk} \odot (\mathbf{X}_{-n} * \boldsymbol{d}_k + b_{nk} \|\boldsymbol{d}_k\|_2^2 \mathbf{W}_{nk}) \end{aligned}$$

where both of $\boldsymbol{\Sigma}_{nk}$ and $\boldsymbol{\mu}_{nk}$ have the same size with $\mathbf{W}_{nk}$. The symbol $\odot$ is the element-wise product operator and $\oslash$ the element-wise division operator. To sample $\boldsymbol{d}_k$, we calculate the posterior mean and covariance for each dictionary element as $\boldsymbol{\Lambda}_k = \mathbf{1} \oslash (\sum_{n=1}^{N} \gamma_n b_{nk} \|\mathbf{W}_{nk}\|_2^2 + \beta_k)$ and $\boldsymbol{\xi}_k = \boldsymbol{\Lambda}_k \odot (\sum_{n=1}^{N} b_{nk} \gamma_n (\mathbf{X}_{-n} * \mathbf{W}_{nk} + \boldsymbol{d}_k \|\mathbf{W}_{nk}\|_2^2))$. The remaining Gibbs update equations are relatively standard in Bayesian factor analysis (Zhou et al., 2009).

3. Multilayered/Deep Models

Using the convolutional factor model discussed above, we yield an approximation to the posterior distribution of all parameters. We wish to use the inferred parameters to perform a multi-scale convolutional factor model. It is possible to perform inference of all layers simultaneously. However, in practice it is reported in the deep-learning literature (Zeiler et al., 2010; LeCun et al.; Hinton et al.; Jarrett et al., 2009; Ranzato et al., 2006; Vincent et al., 2008; Erhan et al., 2010; Lee et al., 2009a;b; Norouzi et al., 2009) that typically sequential design performs well, and therefore we adopt that approach here. When moving from layer l to layer $l + 1$ in the "deep" model, we employ the maximum-likelihood (ML) set of filter weights from layer l, with which we perform decimation and max-pooling, as discussed next.

3.1. Decimation and Max-Pooling

A "max-pooling" step is applied to each $\mathbf{W}_{nk}$, when moving to the next level in the model, with this employed previously in deep models (Lee et al., 2009a) and in recent related image-processing analysis (Boureau et al., 2010). In max-pooling, each matrix $\mathbf{W}_{nk}$ is divided into a contiguous set of blocks, with each such block of size $n_{MP,y} \times n_{MP,x}$. The matrix $\mathbf{W}_{nk}$ is mapped to $\hat{\mathbf{W}}_{nk}$, with the mth value in $\hat{\mathbf{W}}_{nk}$

corresponding to the largest-magnitude component of $\mathbf{W}_{nk}$ within the mth max-pooling region. Since $\mathbf{W}_{nk}$ is of size $(n_y - n'_y + 1) \times (n_x - n'_x + 1)$, each $\hat{\mathbf{W}}_{nk}$ is a matrix of size $(n_y - n'_y + 1)/n_{MP,y} \times (n_x - n'_x + 1)/n_{MP,x}$, assuming integer divisions.

To go to the second layer in the deep model, let $\hat{K}$ denote the number of $k \in \{1, \ldots, K\}$ for which $b_{nk} \neq 0$ for at least one $n \in \{1, \ldots, N\}$. The $\hat{K}$ corresponding max-pooled images from $\{\hat{\mathbf{W}}_{nk}\}_{k=1,K}$ are stacked to constitute a datacube or tensor, with the tensor associated with image n now becoming the input image at the next level of the model. The max-pooling and stacking is performed for all N images, and then the same form of factor-modeling is applied to them (the original K_c color bands is now converted to $\hat{K}$ *effective* spectral bands at the next level). Model fitting at the second layer is performed analogous to that in (1) or (3).

3.2. Model features and visualization

Assume the hierarchical factor-analysis model discussed above is performed for L layers, and therefore after max-pooling the original image $\mathbf{X}_n$ is represented in terms of L tensors $\{\mathbf{X}_n^{(l)}\}_{l=1,L}$ (with $\hat{K}^{(l)}$ layers or "spectral" bands at layer l, and $\{\mathbf{X}_n^{(0)}\}_{n=1,N}$ correspond to the original images for which $\hat{K}^{(0)} = K_c$).

It is of interest to examine the physical meaning of the associated dictionary elements. Specifically, for $l > 1$, we wish to examine the representation of $\boldsymbol{d}_{ki}^{(l)}$ in layer one, *i.e.*, in the image plane. Dictionary element $\boldsymbol{d}_{ki}^{(l)}$ is an $n_y^{(l)} \times n_x^{(l)} \times \hat{K}^{(l)}$ tensor, representing $\boldsymbol{d}_k^{(l)}$ shifted to the point indexed by i, and used in the expansion of $\mathbf{X}_n^{(l)}$; $n_y^{(l)} \times n_x^{(l)}$ represents the number of spatial pixels in $\mathbf{X}_n^{(l)}$. Let $d_{ki\boldsymbol{p}}^{(l)}$ denote the $\boldsymbol{p}$th pixel in $\boldsymbol{d}_{ki}^{(l)}$, where for $l > 1$ vector $\boldsymbol{p}$ identifies a two-dimensional spatial shift as well as a layer level within the tensor $\mathbf{X}_n^{(l)}$; *i.e.*, $\boldsymbol{p}$ is a three-dimensional vector, with the first two coordinates defining a spatial location in the tensor, and the third coordinate identifying a level k in the tensor.

For $l > 1$, the observation of $\boldsymbol{d}_{ki}^{(l)}$ at level $l-1$ is represented as $\boldsymbol{d}_{ki}^{(l)\rightarrow(l-1)} = \sum_{\boldsymbol{p}} d_{ki\boldsymbol{p}}^{(l)} \boldsymbol{d}_{\boldsymbol{p}}^{(l-1)}$, where $\boldsymbol{d}_{\boldsymbol{p}}^{(l-1)}$ represents a shifted version of one of the dictionary elements at layer $l-1$, corresponding to pixel $\boldsymbol{p}$ in $\boldsymbol{d}_{ki}^{(l)}$.

4. Example Results

While the hierarchical form of the proposed model may appear relatively complicated, the number of parameters that need be set is actually modest. In all examples we set $e = g = 1$, and $c = d = 10^{-6}$. The results are most affected by the choice of b, f, and h, these respectively controlling sparsity on filter usage, the sparsity of the factor scores, and the sparsity of the factor loadings (convolutional filters). These parameters are examined in detail below. Unless explicitly stated, $b = b_1 = b_2 = 10^2$, $f = 10^{-6}$ and $h = 10^{-6}$.

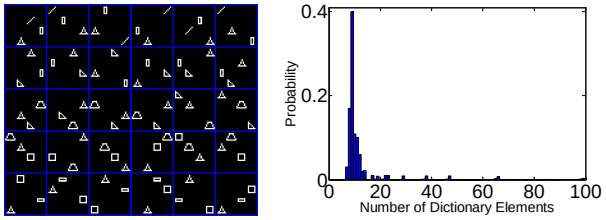

Figure 1. (Left) Images used in synthesized analysis, where each row corresponds one class; (Right) estimated posterior distribution on the number of needed dictionary elements at Layer-2.

4.1. Synthesized data & MNIST examples

To demonstrate the characteristics of the model, we first consider synthesized data. In Figure 1 (left) we generate seven binary canonical shapes, with shifted versions of these basic shapes used to constitute five classes of example images (five "tasks" in the multi-task setting). Each row in the left figure corresponds to one image class, with six images per class (columns). Only the triangle appears in all classes, and specialized shapes are associated with particular classes (*e.g.*, the 45^o line segment is only associated with Class 1, in the first row of the left Figure 1). Each canonical binary shapes is of size 8×8; the thirty synthesized images are also binary, of size 32×32. We consider an HBP analysis, in which each row of the left figure in Figure 1 is one "task".

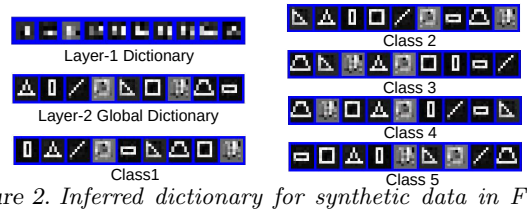

Figure 2. Inferred dictionary for synthetic data in Figure 1(a), based on HBP analysis. From left to right and top to bottom: Layer-1 and Layer-2 global dictionary elements, ordered from left to right based on popularity of use across all images; Inferred dictionary elements at Layer-2 for particular classes/tasks, ranked by class-specific usage.

We consider a two-layer model, with the canonical dictionary elements $\boldsymbol{d}_k^{(l)}$ of spatial size 4×4 ($J = 16$) at layer $l = 1$, and of spatial size 3×3 ($J = 9$) at layer $l = 2$. In all examples, we set the number of dictionary elements at layer one to a relatively small value, as at this layer the objective is to constitute simple primitives (Hinton et al.; Jarrett et al., 2009; Ranzato et al., 2006; Vincent et al., 2008; Lee et al., 2009a;b); here $K = 10$ at layer one. For all higher-level layers

we set K to a relatively large value (here $K = 100$), and allow the HBP construction to *infer* the number of dictionary elements needed and the priority probability of dictionary for each class. The max-pooling ratio is two. To the right in Figure 1 we depict the approximate "global" posterior distribution on filter usage across all tasks (related to the $\{\pi_k\}$ in the HBP model) from the collection samples, for layer two in the model; the distribution is peaked around nine, while as discussed above seven basic shapes were employed to design the toy images. In these examples we employed 30,000 burn-in iterations, and the histogram is based upon 20,000 collection samples. We ran this large number of samples to help insure that the collection samples are representative of the posterior; in practice similar results are obtained with as few as 100 samples, saving significant computational expense. In all subsequent real problem examples including large-size dataset, 1000 burn-in samples were used, with 500 collection samples.

The dictionary elements inferred at layer one and layer two of the model are depicted in Figure 2; in both cases the dictionary elements are projected down to the image plane, and these results are for the ML Gibbs collection sample (to avoid issues with label switching within MCMC, which would undermine showing average dictionary elements). Note that the dictionary elements $\boldsymbol{d}_k^{(1)}$ from layer one constitute basic elements, such as corners, horizontal, vertical and diagonal segments. However, at layer two the dictionary elements $\boldsymbol{d}_k^{(2)}$, when viewed on the image plane, look like the fundamental shapes in Figure 2 used to constitute the synthesized images. In Figure 2 the inferred dictionary elements for each class/task and layer are ordered from left-to-right based, with respect to the frequency with which they are used to represent the "local" task-dependent data, this related to $\{\pi_k^{(t)}\}$ within the HBP. Note, for example, that the 45^o line segment is relatively highly used for class 1 (at layer two in the model), while it is relatively infrequently used for the other tasks.

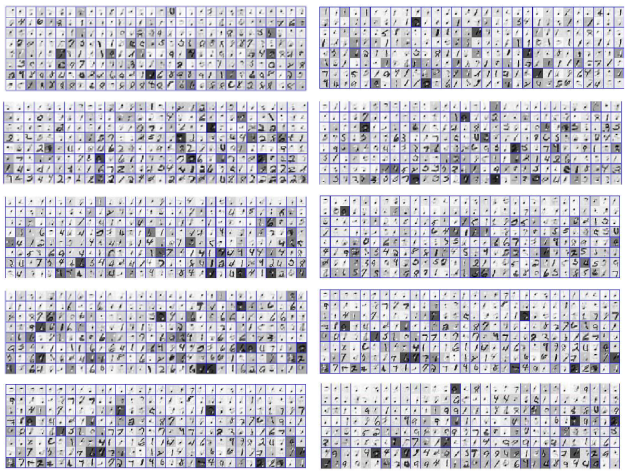

Figure 3. Inferred Layer-2 dictionary for MNIST data. Left: 0, 2, 4 ,6 and 8; Right:1, 3, 5, 7 and 9. The filters are ordered from left-to-right, and then down, based upon usage priority within the class (white is zero).

As a second example, we consider the widely studied MNIST data (http://yann.lecun.com/exdb/mnist/), in which we perform analysis on 5000 images, each 28×28, for digits 0 through 9 (for each digit, we randomly select 500 images). We again consider an HBP analysis, in which now each task/class corresponds to one of the ten digits. We consider a two-layer model, as these images are again relatively simple. In this analysis the dictionary elements at layer one, $\boldsymbol{d}_k^{(1)}$, are 7×7, while the second-layer $\boldsymbol{d}_k^{(2)}$ are 6×6. At layer one a max-pooling ratio of three is employed and $K = 25$, and at layer two $K = 1000$ and a max-pooling ratio of two is used. In Figure 3 we only present the top 240, filters ranked by the usage frequency at layer two for each class. Note that the most prominently used dictionary elements are basic structures, that appear to highlight different canonical strokes within the construction of a particular digit; such simple filters will play an important role in more general images, as discussed next.

4.2. Analysis of Caltech 101 data

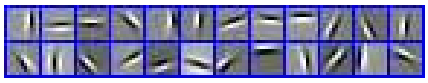

Figure 4. Layer-1 dictionary elements learned for the Caltech 101 dataset.

We next consider the Caltech 101 data set (http://www.vision.caltech.edu/ImageDatasets/Caltech101/), first considering each class of images separately (BP construction of Section 2.1); for these more-sophisticated images, a three-level model is considered, as in (Lee et al., 2009a). When analyzing the Caltech 101 data, we resize each image as 100×100 and use 11×11 Layer-1 filters, meanwhile, the max-pooling ratio is 5. We consider 4×4 Layer-2 filters $\boldsymbol{d}_k^{(2)}$ and 6×6 Layer-3 filters $\boldsymbol{d}_k^{(3)}$. The beta-Bernoulli truncation level, K, was set at 200 and 100 for layers 2 and 3, respectively, and the number of needed filters is inferred via the beta-Bernoulli process. Finally, the max-pooling ratio at layers 2 and 3 is set as 2.

There are 102 image classes in the Caltech 101 data set; we first consider the car class in detail, and then provide a summary exposition on several other image classes (similar class-specific behavior was observed when each of the classes was isolated in isolation). The

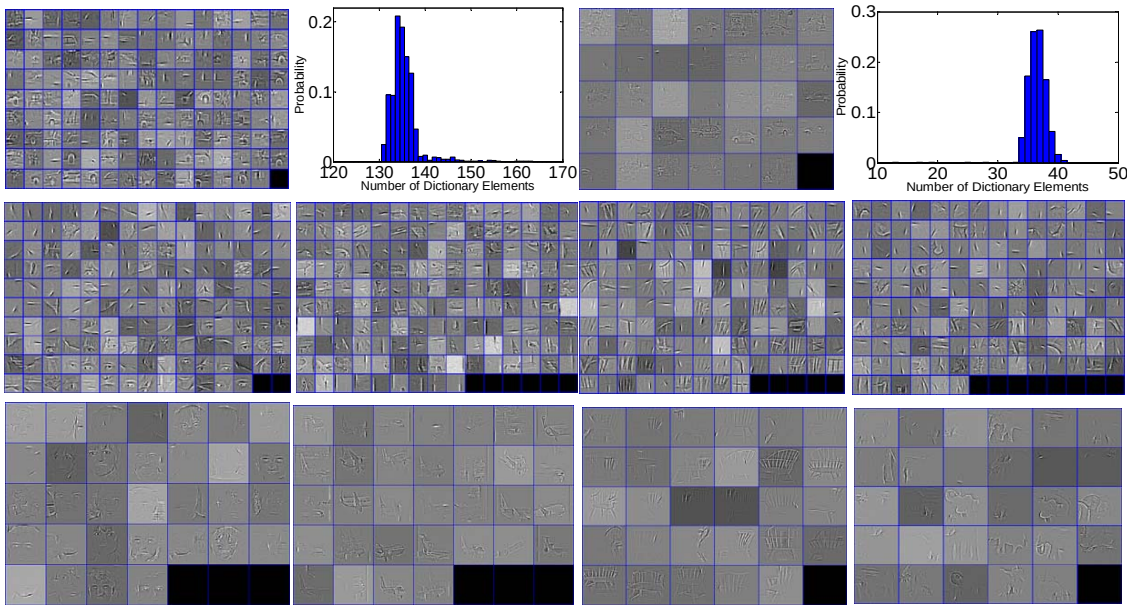

Figure 5. Inferred Layer-2 dictionary for Caltech101 data, BP analysis separately on each class. Row 1: (Right) ordered Layer-2 filters for car class, $\boldsymbol{d}_k^{(2)}$; (Middle-left) approximate posterior distribution on the number of dictionary elements needed for layer two, based upon Gibbs collection samples and car data; (Middle-right) third-layer dictionary elements, $\boldsymbol{d}_k^{(3)}$; (Right) approximate posterior distribution on the number of dictionary elements needed for layer three; Rows 2-3: second-layer and third-layer dictionary elements for face, airplane, chair and elephant classes. Best viewed electronically, zoomed-in.

Layer-1 dictionary elements are depicted in Figure 4, and we focus on $\boldsymbol{d}_k^{(2)}$ and $\boldsymbol{d}_k^{(3)}$, from layers two and three, respectively. Considering the $\boldsymbol{d}_k^{(2)}$ for the car class (in Figure 5), one can observe several parts of cars, and for $\boldsymbol{d}_k^{(3)}$ cars are often clearly visible at Layer-3. Histograms are presented for the approximate posterior distribution of filter usage at Layers 2 and 3; one notes that of the 200 candidate dictionary elements, a mean of roughly 135 of them are used frequently at Layer-2, and 34 are frequently used at Layer-3.

From Figure 5, we observe that when BP is applied to each of the Caltech 101 image classes separately, at Layer-2, and particularly at Layer-3, filters are manifested with structure that looks highly related to the particular image classes (*e.g.*, for the face data, filters that look like eyes at Layer-2, and the sketch of an entire face at Layer-3). Similar filters were observed for single-task learning in (Lee et al., 2009a). However, in Figure 7 we present HBP-learned filters at layers 2, based upon simultaneous analysis of all 102 classes (102 "tasks" within the HBP, with 10 images per task, for a total of 1020 images.); $K = 1000$ in this case (Layer-3 dictionary are put in Supplementary Material due to limit space). The filters are ranked by usage, from left-to-right, and down, and in Figure 7 one observes that the most highly employed HBP-derived filters are characteristic of basic entities at Layer-2. These seem to correspond to basic edge filters, consistent with findings in (Zoran & Weiss, 2009; Puertas et al., 2010); this is also consistent with the basic Layer-2 filters inferred above for the MNIST data. It appears that as the range of image classes considered within an HBP analysis increases, the form of the prominent filters tend toward simple (*e.g.*, edge detection) filter forms.

We also considered HBP results for a fewer number of tasks, and examined the inferred dictionary elements as the number of tasks increased. For example, when simultaneously analyzing ten Caltech 101 classes via the HBP, the inferred dictionary elements at layers 2 and 3 had object-specific structure similar to that above, for single-task BP analysis. As the number of tasks increased beyond 20 classes, the most probable atoms took on a basic, edge-emphasing form, as in Figure 7.

Concerning computation times, considering this task, 200 images in total, Layer-1 with 25 dictionary elements takes 18.3 sec on average per Gibbs sample; Layer-2 with 500 dictionary elements requires 55.2 sec, and Layer-3 with 400 dictionary elements 191.1 sec. All computations were performed in Matlab, on an Intel Core i7 920 2.26GHz with 6GB RAM.

4.3. Sparseness

In deep networks, the ℓ_1-penalty parameter has been utilized to impose sparseness on hidden units (Zeiler et al., 2010; Lee et al., 2009a). However, a detailed examination of the impact of sparseness on various terms of such models has received limited quantitative atten-

tion.

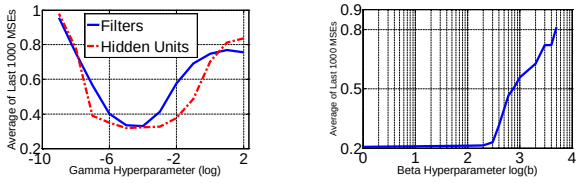

Figure 6. Average MSE calculated from last 1000 Gibbs samples, considering BP analysis (Section 2.1) on the Caltech 101 faces data (averaged across 20 face images considered).

As indicated at the beginning of this section, parameter b controls sparseness on the number of filters employed (via the probability of usage, defined by $\{\pi_k\}$). The normal-gamma prior on the w_{nki} constitutes a Student-t prior, and with $e = 1$, parameter f controls the degree of sparseness imposed on the filter usage (sparseness on the weights $\mathbf{W}_{nk}$). Finally, the components of the filter $\boldsymbol{d}_k$ are also drawn from a Student-t prior, and with $g = 1$, h controls the sparsity of each $\boldsymbol{d}_k$. The above discussion is in terms of the BP construction, for simplicity, while for HBP the parameters b_1 and b_2 play roles analogous to b, with the latter controlling sparseness for specific tasks and the former controlling sparseness across all tasks. For simplicity, below we also focus on the BP construction, and the impact of sparseness parameters b, d and f on sparseness, and model performance.

In Figure 6 we present variation of MSE with these hyperparameters, varying one at a time, and keeping the other fixed as discussed above. These computations were performed on the face Caltech 101 data, averaging 1000 collection samples; 20 face images were considered and averaged over, and similar results were observed using other image classes. A wide range of these parameters yield similar good results, all favoring sparsity (note the axes are on a log scale). Note that as parameter b increases, a more-parsimonious (sparse) use of filters is encouraged, and as b increases the number of inferred dictionary elements (at layer-2 in Figure 8) decreases.

4.4. Classification performance

We address the same classification problem as considered by (Zeiler et al., 2010; Lee et al., 2009a; Lazebnik et al., 2006; Jarrett et al., 2009; Zhang et al., 2006), considering Caltech 101 data (Zeiler et al., 2010; Lee et al., 2009a;b). As in these previous studies, we consider a two-layer model. Two learning paradigms are considered: (i) the dictionary learning is performed using natural scenes, as in (Lee et al., 2009a), with learning via BP; and (ii) the HBP model is employed to learn the filters, where each task corresponds to an image class, as above (the images used for dictionary learning are distinct from those used for classification

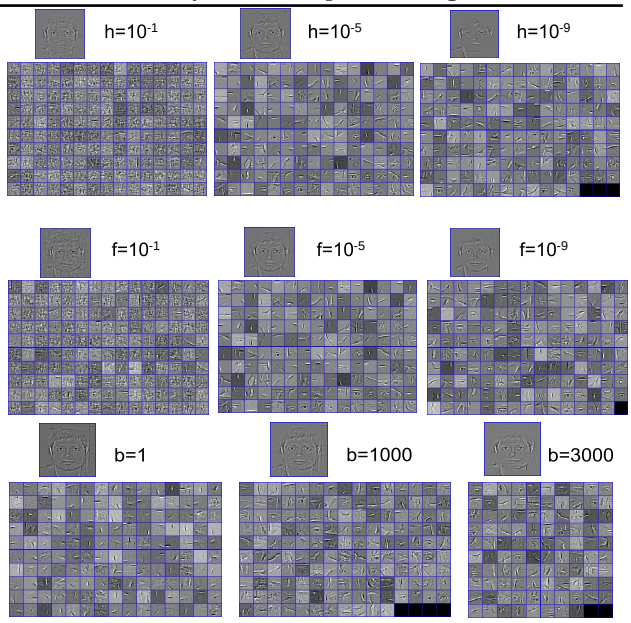

Figure 8. Considering 20 face images from Caltech 101, we examine setting of sparseness parameters; unless otherwise stated, $b = 10^2$, $f = 10^{-5}$ and $h = 10^{-5}$. Parameters h and f are varied in (Top) and (Middle), respectively. In (Bottom), we set $e = 10^{-6}$ and $f = 10^{-6}$ and make hidden units unconstrained to test the influence of parameter b on the model's sparseness. In all of cases, we show the Layer-2 filters (ordered as above) and an example reconstruction.

Table 1. Classification performance of the proposed model, on Caltech 101. The BP results use filters trained with natural-scene data, and the HBP results are based on filters trained using separate Caltech 101 data.

# Training / Testing	15/15	30/30
BP Layer-1	$53.6 \pm 1.5\%$	$62.7 \pm 1.2\%$
BP Layers-1+2	$57.9 \pm 1.4\%$	$65.7 \pm 0.7\%$
HBP Layer-1	$53.5 \pm 1.3\%$	$62.5 \pm 0.8\%$
HBP Layers-1+2	$58.2 \pm 1.2\%$	$65.8 \pm 0.6\%$

testing). Using these feature vectors, we train an SVM as in (Lee et al., 2009a), with results summarized in Table 1. A related table is presented in (Zeiler et al., 2010) for many related models, and our results are very similar to those; our results are most similar to the deep model considered in (Lee et al., 2009a). The results in Table 1 indicate that as the number of classes increases, here to 102 classes, the learned filters at the different layers tend to become generalized, as discussed above. Therefore, classification performance in this case based upon filters learned using independent natural scenes, and the class-dependent filters from the image classes of interest, tend to yield similar classification results. This analysis, based on the novel multi-task HBP construction, confirms this anticipation, implied implicity in the way previous classification tasks of this type have been approached in the deep-learning literature (see aforementioned references).

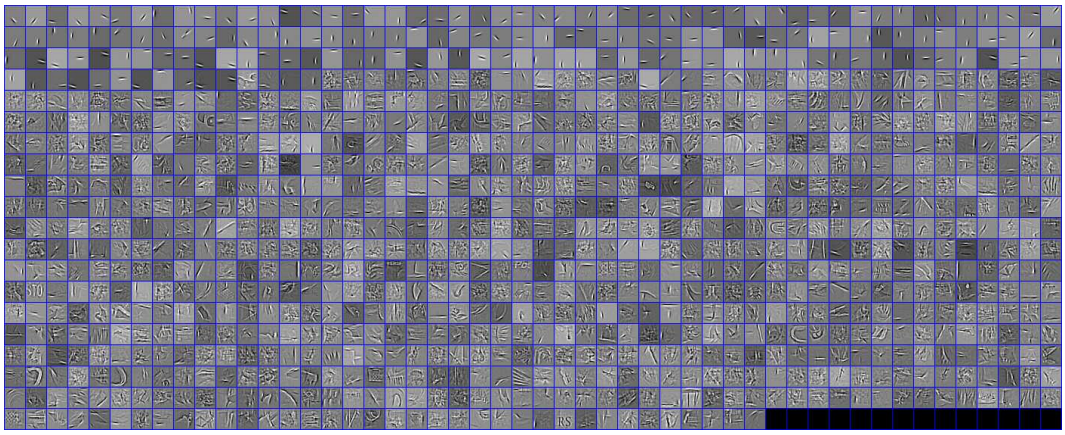

Figure 7. Layer-2 dictionary elements, when HBP analysis performed simultaneously on all 102 image classes in Caltech 101 (102 "tasks"). (Best viewed electronically, zoomed-in).

5. Conclusions

A new convolutional factor analysis model has been developed, and applied to deep feature learning. The model has been implemented using a BP (single-task) and HBP (multi-task) construction, with efficient inference performed using Gibbs analysis. There has also been limited previous work on multi-task deep learning, or on inferring the number of needed filters.

Acknowledgments

The research reported here was supported by AFOSR, ARO, DOE, ONR and NGA.

References

Adams, R.P., Wallach, H.M., and Ghahramani, Z. Learning the structure of deep sparse graphical models. In *AISTATS*, 2010.

Boureau, Y.L., Bach, F., LeCun, Y., and Ponce, J. Learning mid-level features for recognition. In *CVPR*, 2010.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. Why does unsupervised pretraining help deep learning? *JMLR*, 2010.

Griffiths, T. L. and Ghahramani, Z. Infinite latent feature models and the Indian buffet process. In *NIPS*, 2005.

Hinton, G.E., Osindero, S., and Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Computation, 2006*.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, 2010.

Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE, 1998*.

Lee, H., Ekanadham, C., and Ng, A. Y. Sparse deep belief network model for visual area v2. In *NIPS*, 2008.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009a.

Lee, H., Largman, Y., Pham, P., and Ng, A.Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, 2009b.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *ICML*, 2009.

Norouzi, M., Ranjbar, M., and Mori, G. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *CVPR*, 2009.

Paisley, J. and Carin, L. Nonparametric factor analysis with beta process priors. In *ICML*, 2009.

Puertas, G., Bornschein, J., and Lucke, J. The maximal causes of natural scenes are edge filters. In *NIPS*, 2010.

Ranzato, M., Poultney, C.S., Chopra, S., and LeCun, Y. Efficient learning of sparse representations with an energy-based model. In *NIPS*, 2006.

Thibaux, R. and Jordan, M.I. Hierarchical beta processes and the Indian buffet process. In *AISTATS*, 2007.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

Zeiler, M.D., Krishnan, D., Taylor, G.W., and Fergus, R. Deconvolution networks. In *CVPR*, 2010.

Zhang, H., Berg, A. C., Maire, M., and Malik, J. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

Zhou, M., Chen, H., Paisley, J., Ren, L., Sapiro, G., and Carin, L. Non-parametric Bayesian dictionary learning for sparse image representations. In *NIPS*, 2009.

Zoran, D. and Weiss, Y. The tree-dependent components of natural images are edge filters. In *NIPS*, 2009.