

# Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks

By Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng

## Abstract

There has been much interest in unsupervised learning of hierarchical generative models such as deep belief networks (DBNs); however, scaling such models to full-sized, high-dimensional images remains a difficult problem. To address this problem, we present the *convolutional deep belief network*, a hierarchical generative model that scales to realistic image sizes. This model is translation-invariant and supports efficient bottom-up and top-down probabilistic inference. Key to our approach is *probabilistic max-pooling*, a novel technique that shrinks the representations of higher layers in a probabilistically sound way. Our experiments show that the algorithm learns useful high-level visual features, such as object parts, from unlabeled images of objects and natural scenes. We demonstrate excellent performance on several visual recognition tasks and show that our model can perform hierarchical (bottom-up and top-down) inference over full-sized images.

## 1. INTRODUCTION

Machine learning has been highly successful in tackling many real-world artificial intelligence and data mining problems, such as optical character recognition, face detection, autonomous car driving, data mining of biological data, and Web search/information retrieval. However, the success of machine learning systems often requires a large amount of labeled data (which is expensive to obtain) and significant manual feature engineering. These feature representations are often hand-designed, require significant amounts of domain knowledge and human labor, and do not generalize well to new domains. Therefore, it is desirable to be able to develop feature representations automatically while using a small amount of labeled data.

Given these issues, we consider the problem of learning feature representations from unlabeled data, which we call *unsupervised feature learning*. Here, we are interested in primarily using unlabeled data because we can easily obtain a virtually unlimited amount of unlabeled data via the Internet. In fact, even though we do not have labels, there often exist rich structures in unlabeled data. For example, if we look at images of a specific object (e.g., a face), we can easily discover high-level structures such as object parts (e.g., face parts). Given natural images, we

may be able to discover low-level structures such as edges, as well as high-level structures such as corners, local curvatures, and shapes. The main assumption of unsupervised feature learning is that such structures in unlabeled data can be useful in machine learning tasks. For example, if the input data have structures generated from specific object classes (e.g., cars vs. faces), then discovering class-specific patterns (e.g., car wheels or face parts) will be useful for classification, possibly combined with a small amount of labeled data. Similarly, even simple image features (e.g., edges or corners) learned from unlabeled natural images can be useful for object recognition tasks that deal with completely unrelated images. In this context, how can we discover such useful high-level features from unlabeled data?

In recent years, “deep learning” approaches have gained significant interest as a way of building hierarchical representations from unlabeled data.<sup>2, 10, 15, 26, 28</sup> Deep architectures attempt to learn hierarchical structures and seem promising in learning simple concepts first and then successfully building up more complex concepts by composing the simpler ones together. Specifically, deep architectures consist of feature detector units arranged in layers. Lower layers detect simple features and feed into higher layers, which in turn detect more complex features. In particular, the DBN<sup>10</sup> is a multilayer generative model where each layer encodes statistical dependencies among the units in the layer below, and it can be trained to (approximately) maximize the likelihood of its training data. DBNs have been successfully used to learn high-level structures in a wide variety of domains, including handwritten digits<sup>10</sup> and human motion capture data.<sup>31</sup> We build upon the DBN in this paper because we are interested in learning a generative model of images that can be trained in a purely unsupervised manner.

While DBNs have been successful in controlled domains, scaling them to realistic-sized (e.g.,  $200 \times 200$  pixel) images remains challenging for two reasons. First, images are high-dimensional, so the algorithms must scale gracefully and be

A previous version of this paper appeared in *Proceedings of the 26th International Conference on Machine Learning* (Montreal, Canada, 2009).

computationally tractable even when applied to large images. Second, objects can appear at arbitrary locations in images; thus, it is desirable that representations be invariant at least to local translations of the input. We address these issues by incorporating translation invariance. Like LeCun et al.<sup>17</sup> and Grosse et al.,<sup>7</sup> our algorithm learns feature detectors shared among all locations in an image because a feature detector that captures useful information in one part of an image can pick up the same information elsewhere. Thus, our model can represent large images using a small number of feature detectors.

This paper presents the *convolutional deep belief network*, a hierarchical generative model that scales to full-sized images. We also present *probabilistic max-pooling*, a novel technique that allows higher-layer units to cover larger areas of the input in a probabilistically sound way. To the best of our knowledge, ours is the first unsupervised, translation-invariant deep learning model that scales to realistic image sizes and supports full probabilistic inference. The first, second, and third layers of our network learn edge detectors, object parts, and objects, respectively. We show that these representations achieve excellent performance on several visual recognition tasks and allow hidden object parts to be inferred from high-level object information.

## 2. PRELIMINARIES

### 2.1. Restricted Boltzmann machines

In this section, we briefly review the restricted Boltzmann machine (RBM) and DBN models.

The RBM is a two-layer, bipartite, undirected graphical model<sup>a</sup> with a set of binary hidden random variables (units)  $\mathbf{h}$  of dimension  $K$ , a set of (binary or real-valued) visible random variables (units)  $\mathbf{v}$  of dimension  $D$ , and symmetric connections between these two layers represented by a weight matrix  $W \in \mathbb{R}^{D \times K}$ . (See Figure 1 for an illustration of the RBM.) Intuitively, the RBM can be viewed as a Markov Random Field that tries to represent the input data (visible units) with latent factors (hidden units). Here, the weights encode a statistical relationship between the hidden nodes and visible nodes. For example, the weights between the  $j$ th hidden node ( $h_j$ ) and all visible nodes are denoted as  $j$ th “basis” vector, and  $h_j$  are assigned to 1 with high probability whenever the input data match the  $j$ th basis vector (see Equation 4). The formal probabilistic semantics for an RBM is defined by its energy function as follows:

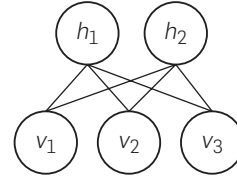
$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})), \quad (1)$$

where  $Z$  is a normalization constant. If the visible units are binary valued, the energy function can be defined as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^D \sum_{j=1}^K v_i W_{ij} h_j - \sum_{j=1}^K b_j h_j - \sum_{i=1}^D c_i v_i, \quad (2)$$

<sup>a</sup> See Koller and Friedman<sup>14</sup> for background on undirected graphical models. In short, the undirected graphical models denote probabilistic models whose joint probability can be written as the product of non-negative potential functions, as in Equations 1–3.

**Figure 1. An example RBM with three visible units ( $D = 3$ ) and two hidden units ( $K = 2$ ). See text for details.**



where  $b_j$  are hidden unit biases ( $\mathbf{b} \in \mathbb{R}^K$ ) and  $c_i$  are visible unit biases ( $\mathbf{c} \in \mathbb{R}^D$ ). If the visible units are real-valued, we can define the energy function by adding a quadratic term to make the distribution well defined:

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2} \sum_{i=1}^D v_i^2 - \sum_{i=1}^D \sum_{j=1}^K v_i W_{ij} h_j - \sum_{j=1}^K b_j h_j - \sum_{i=1}^D c_i v_i. \quad (3)$$

The above energy function defines a joint probability distribution and conditional probability distribution. From the energy function, it is clear that the hidden units are conditionally independent of one another given the visible layer, and vice versa. In particular, the units of a binary hidden layer (conditioned on the visible layer) are independent Bernoulli random variables as follows:

$$P(h_j = 1 | \mathbf{v}) = \sigma \left( \sum_i W_{ij} v_i + b_j \right), \quad (4)$$

where  $\sigma(s) = \frac{1}{1 + \exp(-s)}$  is the sigmoid function. Similarly, if the visible layer is binary-valued, the visible units (conditioned on the hidden layer) are independent Bernoulli random variables as follows:

$$P(v_i = 1 | \mathbf{h}) = \sigma \left( \sum_j W_{ij} h_j + c_i \right). \quad (5)$$

If the visible layer is real-valued, the visible units (conditioned on the hidden layer) are independent Gaussians with diagonal covariance as follows:

$$P(v_i | \mathbf{h}) = \mathcal{N} \left( \sum_j W_{ij} h_j + c_i, 1 \right), \quad (6)$$

where  $\mathcal{N}(\cdot, \cdot)$  is a Gaussian distribution. Therefore, we can perform efficient block Gibbs sampling by alternately sampling each layer’s units (in parallel) given the other layer. We will often refer to a unit’s expected value as its *activation*.

The RBM is a generative model, so, in principle, its parameters can be optimized by performing stochastic gradient descent on the log-likelihood of training data. Unfortunately, computing the exact gradient of the log-likelihood is intractable. Instead, one typically uses the contrastive divergence approximation,<sup>8</sup> which has been shown to work well in practice.

### 2.2. Deep belief networks

The RBM by itself is limited in what it can represent. Its real power emerges when RBMs are stacked to form a DBN, a generative model consisting of many layers. In a DBN,

each layer comprises a set of binary or real-valued units. Two adjacent layers have a full set of connections between them, but no two units in the same layer are connected. Hinton et al.<sup>10</sup> proposed an efficient algorithm for training DBNs, by greedily training each layer (from lowest to highest) as an RBM using the previous layer's activations as inputs.

For example, once a layer of the network is trained, the parameters  $W_{ij}$ ,  $b_j$ ,  $c_i$ 's are frozen and the hidden unit values (given the data) are inferred. These inferred values serve as the input data used to train the next higher layer in the network. Hinton et al.<sup>10</sup> showed that by repeatedly applying such a procedure, one can learn a multilayered DBN. In some cases, this iterative greedy algorithm can be shown to be optimizing a variational lower-bound on the data likelihood, if each layer has at least as many units as the layer below. This greedy layer-wise training approach has been shown to provide a good initialization for parameters for the multilayered network.

### 3. ALGORITHM

Both RBMs and DBNs ignore the 2D structure of images, so weights that detect a given feature must be learned separately for each location. This redundancy makes it difficult to scale these models to full images. One possible way of scaling up is to use massive parallel computation, such as using GPUs, as shown in Raina et al.<sup>25</sup> However, this method may still suffer from having a huge number of parameters. In this section, we present a new method that scales up DBNs using weight-sharing. Specifically, we introduce our model, the convolutional DBN (CDBN), where weights are shared among all locations in an image. This model scales well because inference can be done efficiently using convolution.

#### 3.1. Notation

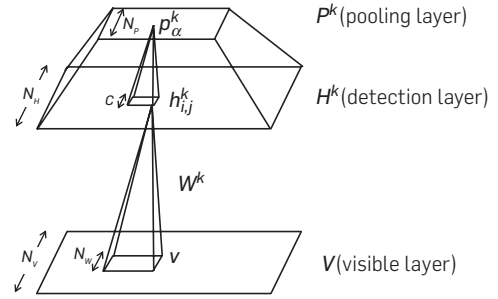
For notational convenience, we will make several simplifying assumptions. First, we assume that all inputs to the algorithm are  $N_v \times N_v$  images, even though there is no requirement that the inputs be square, equally sized, or even 2D. We also assume that all units are binary-valued, while noting that it is straightforward to extend the formulation to the real-valued visible units (see Section 2.1). We use  $*$  to denote convolution,<sup>b</sup> and  $\bullet$  to denote an element-wise product followed by summation, i.e.,  $A \bullet B = \text{tr } A^T B$ . We place a tilde above an array ( $\tilde{A}$ ) to denote flipping the array horizontally and vertically.

#### 3.2. Convolutional RBM

First, we introduce the convolutional RBM (CRBM). Intuitively, the CRBM is similar to the RBM, but the weights between the hidden and visible layers are shared among all locations in an image. The basic CRBM consists of two layers: an input layer  $V$  and a hidden layer  $H$  (corresponding to the lower two layers in Figure 2). The input layer consists of

<sup>b</sup> The convolution of an  $m \times m$  array with an  $n \times n$  array ( $m > n$ ) may result in an  $(m+n-1) \times (m+n-1)$  array (full convolution) or an  $(m-n+1) \times (m-n+1)$  array (valid convolution). Rather than inventing a cumbersome notation to distinguish between these cases, we let it be determined by context.

**Figure 2. Convolutional RBM with probabilistic max-pooling.** For simplicity, only group  $k$  of the detection layer and the pooling layer are shown. The basic CRBM corresponds to a simplified structure with only visible layer and detection (hidden) layer. See text for details.



an  $N_v \times N_v$  array of binary units. The hidden layer consists of  $K$  groups, where each group is an  $N_H \times N_H$  array of binary units, resulting in  $N_H^2 K$  hidden units. Each of the  $K$  groups is associated with a  $N_w \times N_w$  filter ( $N_w \triangleq N_v - N_H + 1$ ); the filter weights are shared across all the hidden units within the group. In addition, each hidden group has a bias  $b_k$  and all visible units share a single bias  $c$ .

We define the energy function  $E(\mathbf{v}, \mathbf{h})$  as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^K \sum_{i,j=1}^{N_H} \sum_{r,s=1}^{N_w} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} - \sum_{k=1}^K b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c \sum_{i,j=1}^{N_v} v_{ij}. \quad (7)$$

Using the operators defined previously,

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^K h^k \bullet (\tilde{W}^k * \mathbf{v}) - \sum_{k=1}^K b_k \sum_{i,j} h_{i,j}^k - c \sum_{i,j} v_{ij}. \quad (8)$$

As with standard RBMs (Section 2.1), we can perform block Gibbs sampling using the following conditional distributions:

$$P(h_{ij}^k = 1 | \mathbf{v}) = \sigma((\tilde{W}^k * \mathbf{v})_{ij} + b_k), \quad (9)$$

$$P(v_{ij} = 1 | \mathbf{h}) = \sigma\left(\left(\sum_k W^k * h^k\right)_{ij} + c\right), \quad (10)$$

where  $\sigma(\cdot)$  is the sigmoid function.<sup>c</sup> Gibbs sampling forms the basis of our inference and learning algorithms.

#### 3.3. Probabilistic max-pooling

To learn high-level representations, we stack CRBMs into a multilayer architecture analogous to DBNs. This architecture is based on a novel operation that we call *probabilistic max-pooling*.

In general, higher-level feature detectors need information from progressively larger input regions. Existing

<sup>c</sup> For the case of real-valued visible units, we can follow the standard formulation as in Section 2.1 and show that

$$P(v_{ij} | \mathbf{h}) = \mathcal{N}\left(\sum_k (W^k * h^k)_{ij} + c, 1\right). \quad (11)$$

translation-invariant representations (e.g., convolutional networks) often involve two kinds of alternating layers: “detection” layers, where responses are computed by convolving a feature detector with the previous layer, and “pooling” layers, which shrink the representation of the detection layers by a constant factor. More specifically, each unit in a pooling layer computes the maximum activation of the units in a small region of the detection layer. Shrinking the representation with max-pooling allows higher-layer representations to be invariant to small translations of the input and reduces the computational burden.

Max-pooling was intended only for deterministic and feed-forward architectures,<sup>17</sup> and it is difficult to perform probabilistic inference (e.g., computing posterior probabilities) since max-pooling is a deterministic operator. In contrast, we are interested in a *generative* model of images that supports full probabilistic inference. Hence, we designed our generative model so that inference involves max-pooling-like behavior.

To simplify the notation, we consider a model with a visible layer  $V$ , a detection layer  $H$ , and a pooling layer  $P$ , as shown in Figure 2. The detection and pooling layers both have  $K$  groups of units, and each group of the pooling layer has  $N_p \times N_p$  binary units. For each  $k \in \{1, \dots, K\}$ , the pooling layer  $P^k$  shrinks the representation of the detection layer  $H^k$  by a factor of  $C$  along each dimension, where  $C$  is a small integer such as 2 or 3. In other words, the detection layer  $H^k$  is partitioned into blocks of size  $C \times C$ , and each block  $\alpha$  is connected to exactly one binary unit  $p_\alpha^k$  in the pooling layer (i.e.,  $N_p = N_H/C$ ). Formally, we define  $B_\alpha \triangleq \{(i, j) : h_{ij}$  belongs to the block  $\alpha\}$ .

The detection units in the block  $B_\alpha$  and the pooling unit  $p_\alpha$  are connected in a single potential which enforces the following constraints: at most one of the detection units may be on, and the pooling unit is on if and only if a detection unit is on. By adding this constraint, we can efficiently sample from the network without explicitly enumerating all  $2^{C^2}$  configurations, as we show later. With this constraint, we can consider these  $C^2 + 1$  units as a single (softmax) random variable which may take on one of  $C^2 + 1$  possible values: one value for each of the detection units being on, and one value indicating that all units are off.

We formally define the energy function of this simplified probabilistic max-pooling-CRBM as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_k \sum_{i,j} (h_{i,j}^k (\tilde{W}^k * v)_{i,j} + b_k h_{i,j}^k) - c \sum_{i,j} v_{i,j} \quad (12)$$

subject to  $\sum_{(i,j) \in B_\alpha} h_{i,j}^k \leq 1, \quad \forall k, \alpha.$

We now discuss sampling the detection layer  $H$  and the pooling layer  $P$  given the visible layer  $V$ . Note that hidden units in group  $k$  receive the following bottom-up signal from layer  $V$ :

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * v)_{ij}. \quad (13)$$

Now, we sample each block independently as a multinomial function of its inputs. Suppose  $h_{i,j}^k$  is a hidden unit contained in block  $\alpha$  (i.e.,  $(i, j) \in B_\alpha$ ), the increase in energy caused by

turning on unit  $h_{i,j}^k$  is  $-I(h_{i,j}^k)$ , and the conditional probability is given by

$$P(h_{i,j}^k = 1 | \mathbf{v}) = \frac{\exp(I(h_{i,j}^k))}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k))} \quad (14)$$

$$P(p_\alpha^k = 0 | \mathbf{v}) = \frac{1}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k))}. \quad (15)$$

In our implementation, we sample the random variables  $\{h_{i,j}^k\}$  and  $p_\alpha^k$  in each block  $\alpha$  from a multinomial distribution, and this can be done in parallel since the blocks are disjoint (i.e., each hidden unit belongs to only one block). Sampling the visible layer  $V$  given the hidden layer  $H$  can be performed in the same way as described in Section 3.2 (e.g., Equation 10 or 11).

### 3.4. Training via sparsity regularization

Our model is overcomplete in that the size of the representation is much larger than the size of the inputs. In fact, since the first hidden layer of the network contains  $K$  groups of units, each roughly the size of the image, it is overcomplete roughly by a factor of  $K$ . In general, overcomplete models run the risk of learning trivial solutions, such as feature detectors representing single pixels. One common solution is to force the representation to be “sparse,” meaning only a tiny fraction of the units should be active in relation to a given stimulus. Following Lee et al.,<sup>18</sup> we regularize the objective function (log-likelihood) to encourage each hidden unit group to have a mean activation close to a small constant. Specifically, we find that the following simple update (followed by contrastive divergence update) works well in practice:

$$\Delta B_k^{sparsity} \propto p - \frac{1}{N_H^2} \sum_{i,j} P(h_{ij}^k = 1 | \mathbf{v}), \quad (16)$$

where  $p$  is a target sparsity, and each image is treated as a mini-batch. The learning rate for sparsity update is chosen as a value that makes the hidden group’s average activation (over the entire training data) close to the target sparsity, while allowing variations depending on specific input images. The overall training algorithm for the convolutional RBM (with probabilistic max-pooling) is described in Algorithm 1.<sup>d</sup>

### 3.5. Convolutional deep belief network

Finally, we are ready to define the CDBN, our hierarchical generative model for full-sized images. Analogous to DBNs, this architecture consists of several max-pooling-CRBMs stacked on top of one another. The network defines an energy function by summing the energy functions for all the individual pairs of layers. Training is accomplished with the same greedy, layer-wise procedure described in Section 2.2: once a given layer is trained, its weights are frozen, and its activations are used as input for the next layer. There is one technical point about learning the biases for each intermediate hidden layer.

<sup>d</sup> To reduce the variance, we followed Hinton and Salakhutdinov<sup>11</sup> by setting  $V^{n+1} = \mathbb{E}_{p[V|H^{(n-1)}]}[V|H^{(n-1)}]$ ; also, we used 1-step CD ( $N_{cd} = 1$ ).

---

**Algorithm 1** A training algorithm for the convolutional RBM

---

**repeat** {over the training data (e.g., a set of training images)}  
 Set  $V^{(0)} := V$  (e.g., set the current image as a mini-batch)  
 Compute the posterior  $Q^{(0)} \triangleq P(H|V^{(0)})$  (Equations 14 and 15).

Sample  $H^{(0)}$  from  $Q^{(0)}$ .

**for**  $n = 1$  to  $N_{cd}$  **do**

Sample  $V^n$  from  $P(V|H^{(n-1)})$  (Equation 10 or 11).<sup>c</sup>

Compute the posterior  $Q^{(n)} \triangleq P(H|V^n)$  (Equations 14 and 15).

Sample  $H^{(n)}$  from  $Q^{(n)}$ .

**end for**

Update weights and biases with contrastive divergence and sparsity regularization:

$$\Delta W^k \propto \frac{1}{N_H^2} (\tilde{Q}^{(0),k} * V^{(0)} - \tilde{Q}^{(n),k} * V^{(n)}) \quad (17)$$

$$\Delta b_k \propto \frac{1}{N_H^2} \sum_{ij} (Q_{ij}^{(0),k} - Q_{ij}^{(n),k}) + \Delta b_k^{\text{sparsity}} \quad (18)$$

$$\Delta c \propto \frac{1}{N_v^2} \sum_{ij} (V_{ij}^{(0)} - V_{ij}^{(n)}) \quad (19)$$

**until** convergence

---

Specifically, the biases of a given layer are learned twice: once when the layer is treated as the “hidden” layer of the CRBM (using the lower layer as visible units), and once when it is treated as the “visible” layer (using the upper layer as hidden units). We resolved this problem by simply fixing the biases with the learned hidden biases in the former case (i.e., using only the biases learned when treating the given layer as the hidden layer of the CRBM). However, we note that a potentially better solution would be to jointly train all the weights for the entire CDBN, using the greedily trained weights as the initialization (e.g., Hinton et al.<sup>10, 29</sup>).

### 3.6. Hierarchical probabilistic inference

Once the parameters have all been learned, we compute the network’s representation of an image by sampling from the joint distribution over all of the hidden layers conditioned on the input image. To sample from this distribution, we use block Gibbs sampling, where each layer’s units are sampled in parallel (see Sections 2.1 and 3.3).

To illustrate the algorithm, we describe a case with one visible layer  $V$ , a detection layer  $H$ , a pooling layer  $P$ , and another, subsequently higher detection layer  $H'$ . Suppose  $H'$  has  $K'$  groups of nodes, and there is a set of shared weights  $\Gamma = \{\Gamma^{1,1}, \dots, \Gamma^{K,K'}\}$  where  $\Gamma^{k,\ell}$  is a weight matrix connecting pooling unit  $P^k$  to detection unit  $H'^{\ell}$ . The definition can be extended to deeper networks in a straightforward way.

Note that an energy function for this sub-network consists of two kinds of potentials: unary terms for each of the groups in the detection layers and interaction terms between  $V$  and  $H$  and between  $P$  and  $H'$ .<sup>c</sup>

<sup>c</sup> To avoid clutter, we removed all the terms that do not depend on  $\mathbf{h}$  and  $\mathbf{p}$ .

$$E(\mathbf{v}, \mathbf{h}, \mathbf{p}, \mathbf{h}') = -\sum_k \mathbf{v} \bullet (W^k * \mathbf{h}^k) - \sum_k b_k \sum_{ij} h_{ij}^k - \sum_{k,\ell} p^k \bullet (\Gamma^{k,\ell} * \mathbf{h}'^{\ell}) - \sum_{\ell} b'_{\ell} \sum_{ij} h'_{ij}^{\ell} \quad (20)$$

To sample the detection layer  $H$  and pooling layer  $P$ , note that the detection layer  $H^k$  receives the following bottom-up signal from layer  $V$ :

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * \mathbf{v})_{ij}, \quad (21)$$

and the pooling layer  $P^k$  receives the following top-down signal from layer  $H'$ :

$$I(p_{\alpha}^k) \triangleq \sum_{\ell} (\Gamma^{k,\ell} * \mathbf{h}'^{\ell})_{\alpha}. \quad (22)$$

Now, we sample each of the blocks independently as a multinomial function of their inputs, as in Section 3.3. If  $(i, j) \in B_{\alpha}$ , the conditional probability is given by

$$P(h_{i,j}^k = 1 | \mathbf{v}, \mathbf{h}') = \frac{\exp(I(h_{i,j}^k) + I(p_{\alpha}^k))}{1 + \sum_{(i',j') \in B_{\alpha}} \exp(I(h_{i',j'}^k) + I(p_{\alpha}^k))} \quad (23)$$

$$P(p_{\alpha}^k = 0 | \mathbf{v}, \mathbf{h}') = \frac{1}{1 + \sum_{(i',j') \in B_{\alpha}} \exp(I(h_{i',j'}^k) + I(p_{\alpha}^k))}. \quad (24)$$

As an alternative to block Gibbs sampling, mean-field (e.g., Salakhutdinov et al.<sup>30</sup>) can be used to approximate the posterior distribution. In all our experiments except for Section 4.5, we used the mean-field approximation to estimate the hidden layer activations given the input.<sup>f</sup>

### 3.7. Discussion

Our model used undirected connections between layers. This approach contrasts with Hinton et al.,<sup>10</sup> which used undirected connections between the top two layers, and top-down directed connections for the layers below. Hinton et al.<sup>10</sup> proposed approximating the posterior distribution using a single bottom-up pass. This feed-forward approach can often effectively estimate the posterior when the image contains no occlusions or ambiguities,<sup>g</sup> but the higher layers cannot help resolve ambiguities in the lower layers. This is due to feed-forward computation, where the lower layer activations are not affected by the higher layer activations. Although Gibbs sampling may more accurately estimate the posterior, applying block Gibbs sampling would be difficult because the nodes in a given layer are not conditionally independent of one another given the layers above and below. In contrast, our treatment using undirected edges enables combining bottom-up and top-down information more efficiently, as shown in Section 4.5.

In our approach, probabilistic max-pooling helps to address scalability by shrinking the higher layers. Moreover, weight-sharing (convolutions) speeds up the algorithm further.

<sup>f</sup> We found that a small number of mean-field iterations (e.g., five iterations) sufficed.

<sup>g</sup> In our experiments, this feed-forward approximation scheme also resulted in similar posteriors of the hidden units and classification performance in most cases.

For example, convolutions between  $K$  filters and an input image are more efficient both in memory and time than repeating  $K N_H^2$  times of inner products between the input image and each of the basis vectors (without weight sharing). As a result, inference in a three-layer network (with  $200 \times 200$  input images) with weight-sharing but without max-pooling is about 10 times slower. Without weight-sharing, it is more than 100 times slower.

In contemporary work that was done independently of ours, Desjardins and Bengio<sup>4</sup> and Norouzi et al.<sup>21</sup> also applied convolutional weight-sharing to RBMs. Our work, however, developed more sophisticated elements such as probabilistic max-pooling to make the algorithm more scalable.

In another contemporary work, Salakhutdinov and Hinton<sup>29</sup> proposed an algorithm to train Boltzmann machines with layer-wise connections (i.e., the same topological structure as in DBNs, but with undirected connections). They called this model the deep Boltzmann machine (DBM). Specifically, they proposed algorithms for pretraining and fine-tuning DBMs. Our treatment of undirected connections is closely related to DBMs. However, our model is different from theirs because we apply convolutional structures and incorporate probabilistic max-pooling into the architecture. Although their work is not convolutional and does not scale to as large images as our model, we note that their pretraining algorithm (a modification of contrastive divergence that duplicates the visible units or hidden units when training the RBMs) or fine-tuning algorithm (joint training of all the parameters using a stochastic approximation procedure<sup>32,35,37</sup>) can also be applied to our model to improve the training procedure.

## 4. EXPERIMENTAL RESULTS

### 4.1. Learning hierarchical representations from natural images

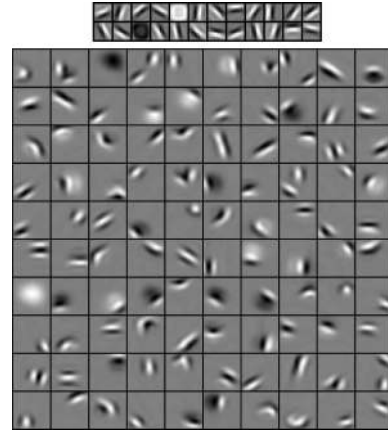
We first tested our model's ability to learn hierarchical representations of natural images. Specifically, we trained a CDBN with two hidden layers from the Kyoto natural image dataset.<sup>h</sup> The first layer consisted of 24 groups (or "bases")<sup>i</sup> of  $10 \times 10$  pixel filters, while the second layer consisted of 100 bases, each one  $10 \times 10$  as well. Since the images were real-valued, we used Gaussian visible units for the first-layer CRBM. The pooling ratio  $C$  for each layer was 2, so the second-layer bases covered roughly twice as large an area as the first-layer bases. We used 0.003 as the target sparsity for the first layer and 0.005 for the second layer.

As Figure 3 (top) shows, the learned first layer bases are oriented, localized edge filters; this result is consistent with much previous work.<sup>1, 9, 22, 23, 28, 33</sup> We note that sparsity regularization during training was necessary to learn these oriented edge filters; when this term was removed, the algorithm failed to learn oriented edges. The learned second layer bases are shown in Figure 3 (bottom), and many of them empirically responded selectively to contours, corners, angles, and surface boundaries in the images. This result is qualitatively consistent with previous work.<sup>12, 13, 18</sup>

<sup>h</sup> Available at [http://www.cnbc.cmu.edu/cplab/data\\_kyoto.html](http://www.cnbc.cmu.edu/cplab/data_kyoto.html)

<sup>i</sup> We will call one hidden group's weights a "basis."

**Figure 3. The first layer bases (top) and the second layer bases (bottom) learned from natural images. Each second layer basis (filter) was visualized as a weighted linear combination of the first layer bases.**



**Table 1. Test classification accuracy for the Caltech-101 data.**

Training size (per class)	15	30
CDBN (first layer)	53.2% $\pm$ 1.2%	60.5% $\pm$ 1.1%
CDBN (first + second layer)	57.7% $\pm$ 1.5%	65.4% $\pm$ 0.5%
Raina et al. <sup>24</sup>	46.6%	—
Ranzato et al. <sup>27</sup>	—	54.0%
Mutch and Lowe <sup>20</sup>	51.0%	56.0%
Lazebnik et al. <sup>16</sup>	54.0%	64.6%
Zhang et al. <sup>38</sup>	59.0% $\pm$ 0.56%	66.2% $\pm$ 0.5%

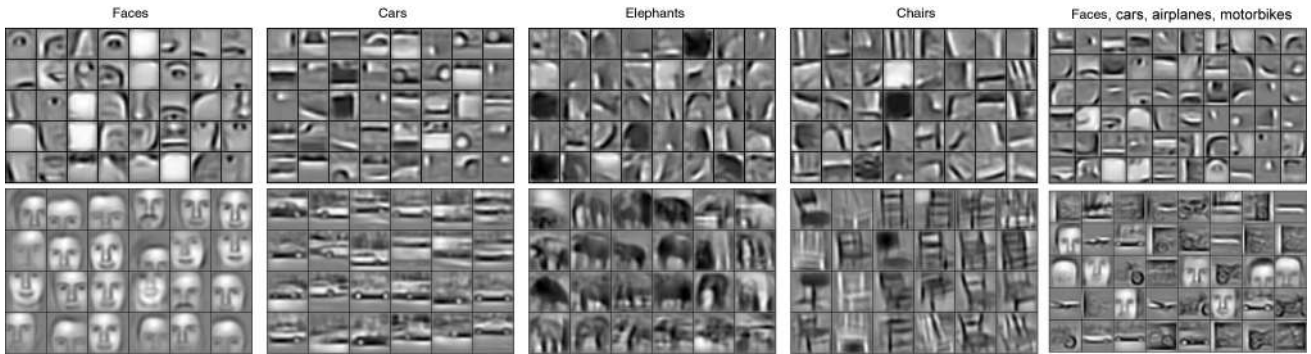
### 4.2. Self-taught learning for object recognition

In the self-taught learning framework,<sup>24</sup> a large amount of unlabeled data can help supervised learning tasks, even when the unlabeled data do not share the same class labels or the same generative distribution with the labeled data. In previous work, sparse coding was used to train single-layer representations from unlabeled data, and the learned representations were used to construct features for supervised learning tasks.

We used a similar procedure to evaluate our two-layer CDBN, described in Section 4.1, on the Caltech-101 object classification task. More specifically, given an image from the Caltech-101 dataset,<sup>5</sup> we scaled the image so that its longer side was 150 pixels and computed the activations of the first and second (pooling) layers of our CDBN. We repeated this procedure after reducing the input image by half and concatenated all the activations to construct features. We used an SVM with a spatial pyramid matching kernel for classification, and the parameters of the SVM were cross-validated. We randomly selected 15 or 30 images per class for training test and testing set, and normalized the result such that classification accuracy for each class was equally weighted (following the standard protocol). We report results averaged over 10 random trials, as shown in Table 1. First, we observe that combining the first and second layers significantly improves the

**Table 2. Test error for MNIST dataset.**

Labeled Training Samples	1,000	2,000	3,000	5,000	60,000
CDBN	2.62% ± 0.12%	2.13% ± 0.10%	1.91% ± 0.09%	1.59% ± 0.11%	0.82%
Ranzato et al. <sup>27</sup>	3.21%	2.53%	—	1.52%	0.64%
Hinton and Salakhutdinov <sup>11</sup>	—	—	—	—	1.20%
Weston et al. <sup>34</sup>	2.73%	—	1.83%	—	1.50%

**Figure 4. Columns 1–4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).**

classification accuracy relative to the first layer alone. Overall, we achieve 57.7% test accuracy using 15 training images per class, and 65.4% test accuracy using 30 training images per class. Our result is competitive with state-of-the-art results using a single type of highly specialized features, such as SIFT, geometric blur, and shape-context.<sup>3, 16, 38</sup> In addition, recall that the CDBN was trained entirely from natural scenes, which are completely unrelated to the classification task. Hence, the strong performance of these features implies that our CDBN learned a highly general representation of images.

We note that current state-of-the-art methods use multiple kernels (or features) together, instead of using a single type of features. For example, Gehler and Nowozin<sup>6</sup> reported a better performance than ours (77.7% for 30 training images/class), but they combined many state-of-the-art features (or kernels) to improve performance. In another approach, Yu et al.<sup>36</sup> used kernel regularization using a (previously published) state-of-the-art kernel matrix to improve the performance of their convolutional neural network model (achieving 67.4% for 30 training examples/class). However, we expect our features can also be used in both settings to further improve performance.

### 4.3. Handwritten digit classification

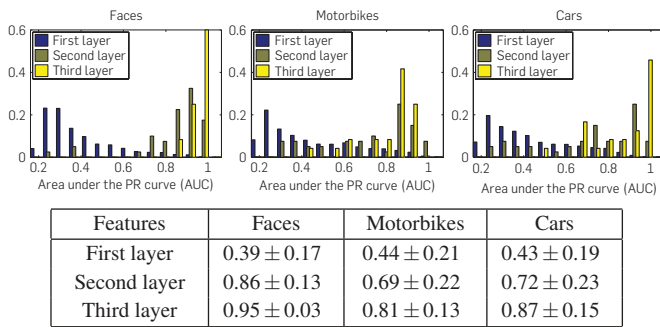
We also evaluated the performance of our model on the MNIST handwritten digit classification task, a widely used benchmark for testing hierarchical representations. We trained 40 first layer bases from MNIST digits, each  $12 \times 12$  pixels, and 40 second layer bases, each  $6 \times 6$ . The pooling ratio  $C$  was 2 for both layers. The first layer bases learned pen-strokes that comprise the digits, and the second layer bases learned bigger digit-parts that combine the pen-strokes. We

constructed feature vectors by concatenating the first and second (pooling) layer activations, and used an SVM for classification using these features. For each labeled training set size, we report the test error averaged over 10 randomly chosen training sets, as shown in Table 2. For the full training set, we obtained 0.8% test error. Our result is comparable to the state of the art.<sup>27</sup>

### 4.4. Unsupervised learning of object parts

We now show that our algorithm can learn hierarchical object-part representations without knowing the position of the objects and the object-parts. Building on the first layer representation learned from natural images, we trained two additional CDBN layers using unlabeled images from single Caltech-101 categories. Training was performed on up to 100 images, and testing was performed on images different than those in the training set. The pooling ratio for the first layer was set as 3. The second layer contained 40 bases, each  $10 \times 10$ , and the third layer contained 24 bases, each  $14 \times 14$ . The pooling ratio in both cases was 2. We used 0.005 as the target sparsity level in both the second and third layers. As shown in Figure 4, the second layer learned features that corresponded to object parts, even though the algorithm was not given any labels that specified the locations of either the objects or their parts. The third layer learned to combine the second layer’s part representations into more complex, higher-level features. Our model successfully learned hierarchical object-part representations of most of the other Caltech-101 categories as well. We note that some of these categories (such as elephants and chairs) have fairly high intra-class appearance variation, due to deformable shapes or different viewpoints. Despite this variation, our model still learns hierarchical, part-based representations fairly robustly.

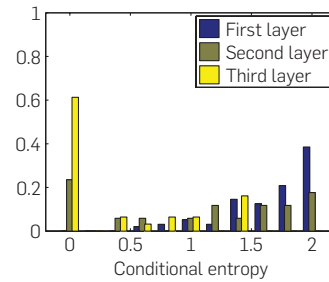
**Figure 5. (top) Histogram of the area under the precision-recall curve (AUC-PR) for three classification problems using class-specific object-part representations. (bottom) Average AUC-PR for each classification problem.**



Higher layers in the CDBN learn features that are not only higher level, but also more specific to particular object categories. We quantitatively measured the specificity of each layer by determining how indicative each individual feature is of object categories. (This setting contrasts with most work in object classification, which focuses on the informativeness of the entire feature set, rather than individual features.) More specifically, we considered three CDBNs trained on faces, motorbikes, and cars, respectively. For each CDBN, we tested the informativeness of individual features from each layer for distinguishing among these three categories. For each feature, we computed the area under the precision-recall curve (larger means more specific). In detail, for any given image, we computed the layer-wise activations using our algorithm, partitioned the activation into  $L \times L$  regions for each group, and computed the  $q\%$  highest quantile activation for each region and each group. If the  $q\%$  highest quantile activation in region  $i$  was  $\gamma$ , we then defined a Bernoulli random variable  $X_{i,L,q}$  with probability  $\gamma$  of being 1. To measure the informativeness between a feature and the class label, we computed the mutual information between  $X_{i,L,q}$  and the class label. We report results using  $(L, q)$  values that maximized the average mutual information (averaging over  $i$ ). Then for each feature, by comparing its values over positive and negative examples, we obtained the precision-recall curve for each classification problem. As shown in Figure 5, the higher-level representations are more selective for the specific object class.

We further tested if the CDBN can learn hierarchical object-part representations when trained on images from several object categories, rather than just one. We trained the second and third layer representations using unlabeled images randomly selected from four object categories (cars, faces, motorbikes, and airplanes). As shown in Figure 4 (far right), the second layer learns class-specific and shared parts, and the third layer learns more object-specific representations. The training examples were unlabeled, so, in a sense, the third layer implicitly clusters the images by object category. As before, we quantitatively measured the specificity of each layer’s individual features to object categories. Since the training was completely unsupervised, whereas the AUC-PR statistic requires knowing which specific

**Figure 6. Histogram of conditional entropy for the representation learned from the mixture of four object classes.**



**Figure 7. Hierarchical probabilistic inference. For each column: (top) input image; (middle) reconstruction from the second layer units after single bottom-up pass, by projecting the second layer activations into the image space; (bottom) reconstruction from the second layer units after 20 iterations of block Gibbs sampling.**



object or object parts the learned bases should represent, we computed the conditional entropy instead. Specifically, we computed the quantile features  $\gamma$  for each layer as previously described, and measured conditional entropy  $H(class | \gamma > 0.95)$ . Informally speaking, conditional entropy measures the entropy of the posterior over class labels when a feature is active. Since lower conditional entropy corresponds to a more peaked posterior, it indicates greater specificity. As shown in Figure 6, the higher-layer features have progressively less conditional entropy, suggesting that they activate more selectively to specific object classes.

#### 4.5. Hierarchical probabilistic inference

Lee and Mumford<sup>19</sup> proposed that the human visual cortex can be modeled conceptually as performing “hierarchical Bayesian inference.” For example, imagine that you observe a face image with its left half in dark illumination, then you would still be able to recognize the face and further infer the darkened parts by combining the image with your prior knowledge of faces. In this experiment, we show that our model can tractably perform such (approximate) hierarchical probabilistic inference in full-sized images. More specifically, we tested the network’s ability to infer the locations of hidden object parts.

To generate examples for evaluation, we used Caltech-101 face images (distinct from the ones the network was trained on). For each image, we simulated an occlusion by zeroing out the left half of the image. We then sampled from the joint posterior over all the hidden layers by performing




Gibbs sampling. Figure 7 shows a visualization of these samples. To ensure that the filling-in required top-down information, we compared with a control condition where only a single upward pass was performed.

In the control (upward-pass only) condition, since there is no evidence from the first layer, the second layer does not respond to the left side. However, with full Gibbs sampling, the bottom-up inputs combine with the context provided by the third layer which has detected the object. This combined evidence significantly improves the second layer representation. Selected examples are shown in Figure 7. Our method may not be competitive to state-of-the-art face completion algorithms using significant prior knowledge and heuristics (e.g., symmetry). However, we find these results promising and view them as a proof of concept for top-down inference.

## 5. CONCLUSION

We presented the CDBN, a scalable generative model for learning hierarchical representations from un-labeled images, and showed that our model performs well in a variety of visual recognition tasks. We believe our approach holds promise as a scalable algorithm for learning hierarchical representations from high-dimensional, complex data.

## Acknowledgments

We give warm thanks to Daniel Oblinger and Rajat Raina for helpful discussions. This work was supported by the DARPA transfer learning program under contract number FA8750-05-2-0249. 

## References

- Bell, A.J., Sejnowski, T.J. The 'independent components' of natural scenes are edge filters. *Vis. Res.* 37, 23 (1997), 3327–3338.
- Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 2007.
- Berg, A.C., Berg, T.L., Malik, J. Shape matching and object recognition using low distortion correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
- Desjardins, G., Bengio, Y. Empirical evaluation of convolutional RBMs for vision. Technical report, University of Montreal, Montreal, Quebec, Canada, 2008.
- Fei-Fei, L., Fergus, R., Perona, P. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative Model Based Vision*, 2004.
- Gehler, P., Nowozin, S. On feature combination for multiclass object classification. In *Proceedings of the International Conference on Computer Vision*, 2009.
- Grosse, R., Raina, R., Kwong, H., Ng, A.Y. Shift-invariant sparse coding for audio classification. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2007.
- Hinton, G.E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14, 8 (2002), 1771–1800.
- Hinton, G.E., Osindero, S., Bao, K. Learning causally linked MRFs. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2005.
- Hinton, G.E., Osindero, S., Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* 18, 7 (2006), 1527–1554.
- Hinton, G.E., Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
- Hyvarinen, A., Gutmann, M., Hoyer, P.O. Statistical model of natural stimuli predicts edge-like pooling of spatial frequency channels in V2. *BMC Neurosci.* 6 (2005), 12.
- Ito, M., Komatsu, H. Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. *J. Neurosci.* 24, 13 (2004), 3313–3324.
- Koller, D., Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA, 2009.
- Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the International Conference on Machine Learning*, 2007.
- Lazebnik, S., Schmid, C., Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
- LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1 (1989), 541–551.
- Lee, H., Ekanadham, C., Ng, A.Y. Sparse deep belief network model for visual area V2. In *Advances in Neural Information Processing Systems*, 2008.
- Lee, T.S., Mumford, D. Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A* 20, 7 (2003), 1434–1448.
- Mutch, J., Lowe, D.G. Multiclass object recognition with sparse, localized features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
- Norouzi, M., Ranjbar, M., Mori, G. Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- Olshausen, B.A., Field, D.J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381 (1993), 607–609.
- Osindero, S., Welling, M., Hinton, G.E. Topographic product models applied to natural scene statistics. *Neural Comput.* 18, 2 (2006), 381–344.
- Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning*, 2007.
- Raina, R., Madhavan, A., Ng, A.Y. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the International Conference on Machine Learning*, 2009.
- Ranzato, M., Boureau, Y., LeCun, Y. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems*, 2007.
- Ranzato, M., Huang, F.-J., Boureau, Y.-L., LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision*
- and *Pattern Recognition*, 2007.
- Ranzato, M., Poultney, C., Chopra, S., LeCun, Y. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems* (2006), 1137–1144, 2006.
- Salakhutdinov, R., Hinton, G.E. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.
- Salakhutdinov, R., Mnih, A., Hinton, G. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning*, 2007.
- Taylor, G.W., Hinton, G.E., Roweis, S.T. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems* 19, 2007.
- Tieleman, T. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine Learning*, 2008.
- van Hateren, J.H., van der Schaaf, A. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. R. Soc. B* 265 (1998), 359–366.
- Weston, J., Ratle, F., Collobert, R. Deep learning via semi-supervised embedding. In *Proceedings of the International Conference on Machine Learning*, 2008.
- Younes, L. Maximum of likelihood estimation for Gibbsian fields. *Probab. Theory Relat. Fields* 82 (1989), 625–645.
- Yu, K., Xu, W., Gong, Y. Deep learning with kernel regularization for visual recognition. In *Advances in Neural Information Processing Systems*, 2009.
- Yuille, A.L. The convergence of contrastive divergences. In *Advances in Neural Information Processing Systems* 17, 2005.
- Zhang, H., Berg, A.C., Maire, M., Malik, J. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

**Honglak Lee** (honglak@eecs.umich.edu), Computer Science and Engineering Division, University of Michigan, Ann Arbor, MI.

**Roger Grosse** (rgrosse@mit.edu), CSAIL, Massachusetts Institute of Technology, Cambridge, MA.

**Rajesh Ranganath and Andrew Y. Ng** ([rajeshr,ang]@cs.stanford.edu), Computer Science Department, Stanford University, Stanford, CA.