Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

Honglak Lee HLLEE@CS.STANFORD.EDU
Roger Grosse RGROSSE@CS.STANFORD.EDU
Rajesh Ranganath RAJESHR@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

There has been much interest in unsupervised learning of hierarchical generative models such as deep belief networks. 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 which 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 which 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

The visual world can be described at many levels: pixel intensities, edges, object parts, objects, and beyond. The prospect of learning hierarchical models which simultaneously represent multiple levels has recently generated much interest. Ideally, such "deep" representations would learn hierarchies of feature detectors, and further be able to combine top-down and bottom-up processing of an image. For instance, lower layers could support object detection by spotting low-level features indicative of object parts. Conversely, information about objects in the higher layers could resolve lower-level ambiguities in the image or infer the locations of hidden object parts.

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. There have been several approaches to learning deep networks (LeCun et al., 1989; Bengio et al., 2006; Ranzato et al., 2006; Hinton et al., 2006). In particular, the deep belief network (DBN) (Hinton et al., 2006) is a multilayer generative model where each layer encodes statistical dependencies among the units in the layer below it; it is trained to (approximately) maximize the likelihood of its training data. DBNs have been successfully used to learn high-level structure in a wide variety of domains, including handwritten digits (Hinton et al., 2006) and human motion capture data (Taylor et al., 2007). We build upon the DBN in this paper because we are interested in learning a generative model of images which can be trained in a purely unsupervised manner.

While DBNs have been successful in controlled domains, scaling them to realistic-sized (e.g., 200x200 pixel) images remains challenging for two reasons. First, images are high-dimensional, so the algorithms must scale gracefully and be 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. (1989) and Grosse et al. (2007), we learn feature detectors which are shared among all locations in an image, because features which capture useful information in one part of an image can pick up the same information elsewhere. Thus, our model can represent large images using only a small number of feature detectors.

This paper presents the *convolutional deep belief network*, a hierarchical generative model that scales to full-sized images. Another key to our approach is

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

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 translation invariant hierarchical generative model which supports both top-down and bottom-up probabilistic inference and scales to realistic image sizes. 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

The restricted Boltzmann machine (RBM) is a two-layer, bipartite, undirected graphical model with a set of binary hidden units $\mathbf{h}$, a set of (binary or real-valued) visible units $\mathbf{v}$, and symmetric connections between these two layers represented by a weight matrix W. The probabilistic semantics for an RBM is defined by its energy function as follows:

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

where Z is the partition function. If the visible units are binary-valued, we define the energy function as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i c_i v_i,$$

where b_j are hidden unit biases and c_i are visible unit biases. If the visible units are real-valued, we can define the energy function as:

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

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 layer (conditioned on the other layer) are independent Bernoulli random variables. If the visible layer is real-valued, the visible units (conditioned on the hidden layer) are Gaussian with diagonal covariance. 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*.

In principle, the RBM parameters can be optimized by performing stochastic gradient ascent 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 (Hinton, 2002), 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 deep belief network, 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. (2006) proposed an efficient algorithm for training deep belief networks, by greedily training each layer (from lowest to highest) as an RBM using the previous layer's activations as inputs. This procedure works well in practice.

3. Algorithms

RBMs and DBNs both ignore the 2-D structure of images, so weights that detect a given feature must be learned separately for every location. This redundancy makes it difficult to scale these models to full images. (However, see also (Raina et al., 2009).) In this section, we introduce our model, the convolutional DBN, whose 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 two-dimensional. 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[1], and $\bullet$ to denote 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 1). The input layer consists of 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

[1]The convolution of an $m \times m$ array with an $n \times n$ array may result in an $(m+n-1) \times (m+n-1)$ array or an $(m-n+1) \times (m-n+1)$ array. Rather than invent a cumbersome notation to distinguish these cases, we let it be determined by context.

($N_W \triangleq N_V - N_H + 1$); 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:

$$\begin{aligned} P(\mathbf{v}, \mathbf{h}) &= \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})) \\ E(\mathbf{v}, \mathbf{h}) &= -\sum_{k=1}^{K} \sum_{i,j=1}^{N_H} \sum_{r,s=1}^{N_W} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} \\ &\quad -\sum_{k=1}^{K} b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c \sum_{i,j=1}^{N_V} v_{ij}. \end{aligned} \tag{1}$$

Using the operators defined previously,

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

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

$$\begin{aligned} P(h_{ij}^k = 1 | \mathbf{v}) &= \sigma((\tilde{W}^k * v)_{ij} + b_k) \\ P(v_{ij} = 1 | \mathbf{h}) &= \sigma((\sum_k W^k * h^k)_{ij} + c), \end{aligned}$$

where σ is the sigmoid function. Gibbs sampling forms the basis of our inference and learning algorithms.

3.3. Probabilistic max-pooling

In order 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 translation-invariant representations, such as convolutional networks, often involve two kinds of layers in alternation: "detection" layers, whose 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 feed-forward architectures. In contrast, we are interested in a *generative* model of images which supports both top-down and bottom-up inference. Therefore, we designed our generative model so that inference involves max-pooling-like behavior.

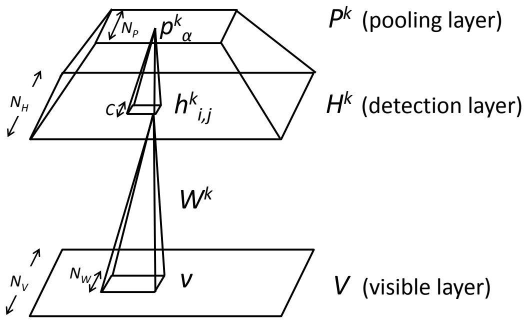


Figure 1. Convolutional RBM with probabilistic max-pooling. For simplicity, only group k of the detection layer and the pooing layer are shown. The basic CRBM corresponds to a simplified structure with only visible layer and detection (hidden) layer. See text for details.

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 1. 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 $k \in \{1, ..., K\}$, 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. I.e., the detection layer H^k is partitioned into blocks of size $C \times C$, and each block α is connected to exactly one binary unit p_α^k in the pooling layer (i.e., $N_P = N_H/C$). Formally, we define $B_\alpha \triangleq \{(i,j) : h_{ij} \text{ belongs to the block } \alpha.\}$.

The detection units in the block B_α and the pooling unit p_α 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. Equivalently, we can consider these C^2+1 units as a single 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:

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}) &= -\sum_k \sum_{i,j} \left(h_{i,j}^k (\tilde{W}^k * v)_{i,j} + b_k h_{i,j}^k \right) - c \sum_{i,j} v_{i,j} \\ \text{subj. to} &\quad \sum_{(i,j) \in B_\alpha} h_{i,j}^k \leq 1, \quad \forall k, \alpha. \end{aligned}$$

We now discuss sampling the detection layer H and the pooling layer P given the visible layer V. Group k receives the following bottom-up signal from layer V:

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * v)_{ij}. \tag{2}$$

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 α (i.e., $(i,j) \in B_\alpha$), the increase in energy caused by turning on unit $h^k_{i,j}$ is $-I(h^k_{i,j})$, and the conditional probability is given by:

$$\begin{aligned} P(h^k_{i,j} = 1|\mathbf{v}) &= \frac{\exp(I(h^k_{i,j}))}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h^k_{i',j'}))} \\ P(p^k_\alpha = 0|\mathbf{v}) &= \frac{1}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h^k_{i',j'}))}. \end{aligned}$$

Sampling the visible layer V given the hidden layer H can be performed in the same way as described in Section 3.2.

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," in that only a tiny fraction of the units should be active in relation to a given stimulus (Olshausen & Field, 1996; Lee et al., 2008). In our approach, like Lee et al. (2008), we regularize the objective function (data log-likelihood) to encourage each of the hidden units to have a mean activation close to some small constant ρ. For computing the gradient of sparsity regularization term, we followed Lee et al. (2008)'s method.

3.5. Convolutional deep belief network

Finally, we are ready to define the convolutional deep belief network (CDBN), our hierarchical generative model for full-sized images. Analogously to DBNs, this architecture consists of several max-pooling-CRBMs stacked on top of one another. The network defines an energy function by summing together the energy functions for all of 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 to the next layer.

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 the units of each layer are sampled in parallel (see Sections 2.1 & 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 $\Gamma = \{\Gamma^{1,1}, \ldots, \Gamma^{K,K'}\}$, where $\Gamma^{k,\ell}$ is a weight matrix connecting pooling unit P^k to detection unit H'^ℓ. 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':

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}, \mathbf{p}, \mathbf{h}') = &-\sum_k v \bullet (W^k * h^k) - \sum_k b_k \sum_{ij} h^k_{ij} \\ &- \sum_{k,\ell} p^k \bullet (\Gamma^{k\ell} * h'^\ell) - \sum_\ell b'_\ell \sum_{ij} h'^\ell_{ij} \end{aligned}$$

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:

$$I(h^k_{ij}) \triangleq b_k + (\tilde{W}^k * v)_{ij}, \qquad (3)$$

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

$$I(p^k_\alpha) \triangleq \sum_\ell (\Gamma^{k\ell} * h'^\ell)_\alpha. \qquad (4)$$

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:

$$\begin{aligned} P(h^k_{i,j} = 1|\mathbf{v}, \mathbf{h}') &= \frac{\exp(I(h^k_{i,j}) + I(p^k_\alpha))}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h^k_{i',j'}) + I(p^k_\alpha))} \\ P(p^k_\alpha = 0|\mathbf{v}, \mathbf{h}') &= \frac{1}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h^k_{i',j'}) + I(p^k_\alpha))}. \end{aligned}$$

As an alternative to block Gibbs sampling, mean-field can be used to approximate the posterior distribution.[2]

3.7. Discussion

Our model used undirected connections between layers. This contrasts with Hinton et al. (2006), which used undirected connections between the top two layers, and top-down directed connections for the layers below. Hinton et al. (2006) proposed approximating the posterior distribution using a single bottom-up pass. This feed-forward approach often can effectively estimate the posterior when the image contains no occlusions or ambiguities, but the higher layers cannot help resolve ambiguities in the lower layers. Although Gibbs sampling may more accurately estimate the posterior in this network, applying block Gibbs sampling would be difficult because the nodes in a given layer

[2]In all our experiments except for Section 4.5, we used the mean-field approximation to estimate the hidden layer activations given the input images. We found that five mean-field iterations sufficed.

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; weight-sharing (convolutions) further speeds up the algorithm. For example, inference in a three-layer network (with 200x200 input images) using weight-sharing but without max-pooling was about 10 times slower. Without weight-sharing, it was more than 100 times slower.

In work that was contemporary to and done independently of ours, Desjardins and Bengio (2008) also applied convolutional weight-sharing to RBMs and experimented on small image patches. Our work, however, develops more sophisticated elements such as probabilistic max-pooling to make the algorithm more scalable.

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.[3] The first layer consisted of 24 groups (or "bases")[4] of 10x10 pixel filters, while the second layer consisted of 100 bases, each one 10x10 as well.[5] As shown in Figure 2 (top), the learned first layer bases are oriented, localized edge filters; this result is consistent with much prior work (Olshausen & Field, 1996; Bell & Sejnowski, 1997; Ranzato et al., 2006). We note that the sparsity regularization during training was necessary for learning 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 2 (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 (Ito & Komatsu, 2004; Lee et al., 2008).

4.2. Self-taught learning for object recognition

Raina et al. (2007) showed that large unlabeled data can help in supervised learning tasks, even when the

[3]`http://www.cnbc.cmu.edu/cplab/data_kyoto.html`

[4]We will call one hidden group's weights a "basis."

[5]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 cover roughly twice as large an area as the first-layer ones.

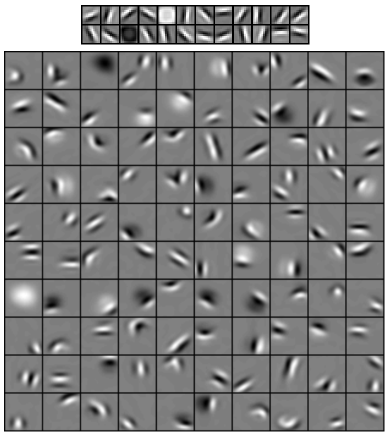


Figure 2. The first layer bases (top) and the second layer bases (bottom) learned from natural images. Each second layer basis (filter) was visualized as a weighted linear combination of the first layer bases.

unlabeled data do *not* share the same class labels, or the same generative distribution, as the labeled data. This framework, where generic unlabeled data improve performance on a supervised learning task, is known as *self-taught learning.* In their experiments, they used sparse coding to train a single-layer representation, and then used the learned representation 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.[6] The results are shown in Table 1. First, we observe that combining the first and second layers significantly improves the 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 *highly-specialized single features*, such as SIFT, geometric blur, and shape-context (Lazebnik et al., 2006; Berg et al., 2005; Zhang et al., 2006).[7] Recall that the CDBN was trained en-

[6]Details: Given an image from the Caltech-101 dataset (Fei-Fei et al., 2004), 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 15/30 training set and 15/30 test set images respectively, 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.

[7]Varma and Ray (2007) reported better performance than ours (87.82% for 15 training images/class), but they combined many state-of-the-art features (or kernels) to improve the performance. In another approach, Yu et al. (2009) used kernel regularization using a (previously published) state-of-the-art kernel matrix to improve the per-

Table 1. Classification accuracy for the Caltech-101 data

Training Size	15	30
CDBN (first layer)	53.2±1.2%	60.5±1.1%
CDBN (first+second layers)	57.7±1.5%	65.4±0.5%
Raina et al. (2007)	46.6%	-
Ranzato et al. (2007)	-	54.0%
Mutch and Lowe (2006)	51.0%	56.0%
Lazebnik et al. (2006)	54.0%	64.6%
Zhang et al. (2006)	59.0±0.56%	66.2±0.5%

tirely 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.

4.3. Handwritten digit classification

We further 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 12x12 pixels, and 40 second layer bases, each 6x6. The pooling ratio C was 2 for both layers. The first layer bases learned "strokes" that comprise the digits, and the second layer bases learned bigger digit-parts that combine the 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 (Ranzato et al., 2007; Weston et al., 2008).[8]

4.4. Unsupervised learning of object parts

We now show that our algorithm can learn hierarchical object-part representations *in an unsupervised setting.* Building on the first layer representation learned from natural images, we trained two additional CDBN layers using unlabeled images from single Caltech-101 categories.[9] As shown in Figure 3, the second layer learned features corresponding to object parts, even though the algorithm was not given any labels specifying 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, our model still learns hierarchical, part-based representations fairly robustly.

Higher layers in the CDBN learn features which are not only higher level, but also more specific to particular object categories. We now quantitatively measure the specificity of each layer by determining how indicative each individual feature is of object categories. (This contrasts with most work in object classification, which focuses on the informativeness of the entire feature set, rather than individual features.) More specifically, we consider three CDBNs trained on faces, motorbikes, and cars, respectively. For each CDBN, we test the informativeness of individual features from each layer for distinguishing among these three categories. For each feature,[10] we computed area under the precision-recall curve (larger means more specific).[11] As shown in Figure 4, 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 3 (far right), the second layer learns class-specific as well as shared parts, and the third layer learns more object-specific representations. (The training examples were unlabeled, so in a sense, this means 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. Because the training was completely unsupervised, whereas the AUC-PR statistic requires knowing which specific object or object parts the learned bases should represent, we instead computed conditional entropy.[12] Informally speaking, conditional entropy measures the entropy of

formance of their convolutional neural network model.

[8]We note that Hinton and Salakhutdinov (2006)'s method is non-convolutional.

[9]The images were unlabeled in that the position of the object is unspecified. Training was on up to 100 images, and testing was on different images than the training set. The pooling ratio for the first layer was set as 3. The second layer contained 40 bases, each 10x10, and the third layer contained 24 bases, each 14x14. The pooling ratio in both cases was 2.

[10]For a given image, we computed the layerwise activations using our algorithm, partitioned the activation into LxL 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 is γ, we then define a Bernoulli random variable $X_{i,L,q}$ with probability γ 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. Results reported are using (L, q) values that maximized the average mutual information (averaging over i).

[11]For each feature, by comparing its values over positive examples and negative examples, we obtained the precision-recall curve for each classification problem.

[12]We computed the quantile features γ for each layer as previously described, and measured conditional entropy $H(class|\gamma > 0.95)$.

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. (2007)	3.21%	2.53%	-	1.52%	0.64%
Hinton and Salakhutdinov (2006)	-	-	-	-	1.20%
Weston et al. (2008)	2.73%	-	1.83%	-	1.50%

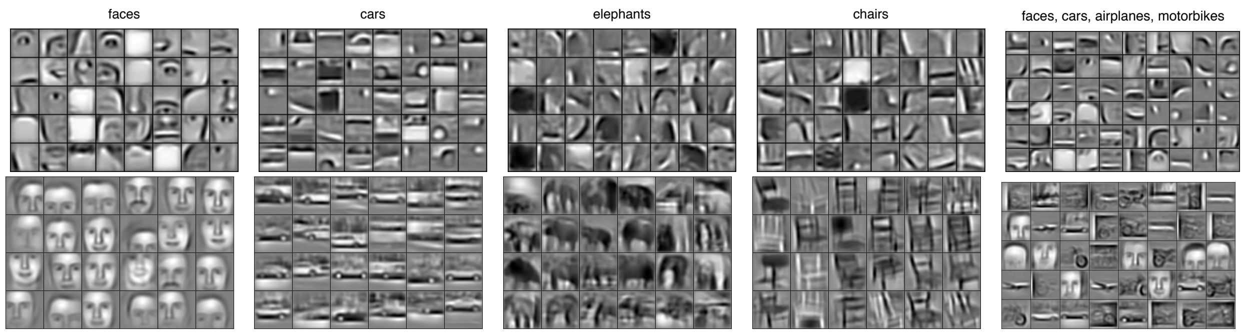
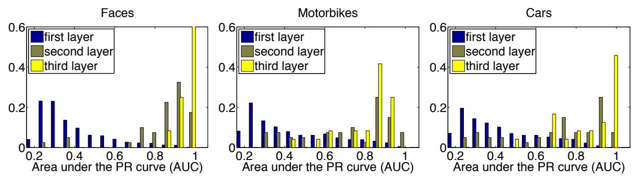


Figure 3. Columns 1-4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).



Features	Faces	Motorbikes	Cars
First layer	0.39±0.17	0.44±0.21	0.43±0.19
Second layer	0.86±0.13	0.69±0.22	0.72±0.23
Third layer	0.95±0.03	0.81±0.13	0.87±0.15

Figure 4. (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.

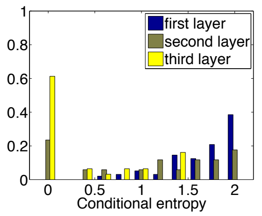


Figure 5. Histogram of conditional entropy for the representation learned from the mixture of four object classes.

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 5, 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 (2003) proposed that the human visual cortex can conceptually be modeled as performing "hierarchical Bayesian inference." For example, if you observe a face image with its left half in dark illumination, you can still 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.

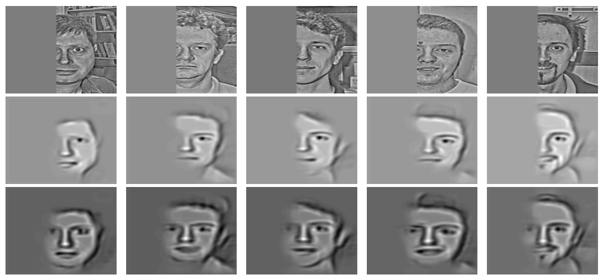


Figure 6. Hierarchical probabilistic inference. For each column: (top) input image. (middle) reconstruction from the second layer units after single bottom-up pass, by projecting the second layer activations into the image space. (bottom) reconstruction from the second layer units after 20 iterations of block Gibbs sampling.

To generate the 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 of the hidden layers by performing Gibbs sampling. Figure 6 shows a visualization of these samples. To ensure that the filling-in required top-down information, we compare 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 much 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 6.

5. Conclusion

We presented the convolutional deep belief network, a scalable generative model for learning hierarchical representations from unlabeled 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.

Acknowledgment

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. (1997). The 'independent components' of natural scenes are edge filters. *Vision Research*, *37*, 3327–3338.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. *Adv. in Neural Information Processing Systems.*

Berg, A. C., Berg, T. L., & Malik, J. (2005). Shape matching and object recognition using low distortion correspondence. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 26–33).

Desjardins, G., & Bengio, Y. (2008). *Empirical evaluation of convolutional RBMs for vision* (Technical Report).

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. *CVPR Workshop on Gen.-Model Based Vision.*

Grosse, R., Raina, R., Kwong, H., & Ng, A. (2007). Shift-invariant sparse coding for audio classification. *Proceedings of the Conference on Uncertainty in AI.*

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

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Hinton, G. E., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

Ito, M., & Komatsu, H. (2004). Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. *J. Neurosci.*, *24*, 3313–3324.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *IEEE Conference on Computer Vision and Pattern Recognition.*

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*, 541–551.

Lee, H., Ekanadham, C., & Ng, A. Y. (2008). Sparse deep belief network model for visual area V2. *Advances in Neural Information Processing Systems.*

Lee, T. S., & Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, *20*, 1434–1448.

Mutch, J., & Lowe, D. G. (2006). Multiclass object recognition with sparse, localized features. *IEEE Conf. on Computer Vision and Pattern Recognition.*

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. *International Conference on Machine Learning* (pp. 759–766).

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. *International Conf. on Machine Learning.*

Ranzato, M., Huang, F.-J., Boureau, Y.-L., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *IEEE Conference on Computer Vision and Pattern Recognition.*

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems* (pp. 1137–1144).

Taylor, G., Hinton, G. E., & Roweis, S. (2007). Modeling human motion using binary latent variables. *Adv. in Neural Information Processing Systems.*

Varma, M., & Ray, D. (2007). Learning the discriminative power-invariance trade-off. *International Conference on Computer Vision.*

Weston, J., Ratle, F., & Collobert, R. (2008). Deep learning via semi-supervised embedding. *International Conference on Machine Learning.*

Yu, K., Xu, W., & Gong, Y. (2009). Deep learning with kernel regularization for visual recognition. *Adv. Neural Information Processing Systems.*

Zhang, H., Berg, A. C., Maire, M., & Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *IEEE Conference on Computer Vision and Pattern Recognition.*