

Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines

Roland Memisevic

roland@cs.toronto.edu

Geoffrey E. Hinton

hinton@cs.toronto.edu

Department of Computer Science, University of Toronto, Toronto M5S 3G4, Canada

To allow the hidden units of a restricted Boltzmann machine to model the transformation between two successive images, Memisevic and Hinton (2007) introduced three-way multiplicative interactions that use the intensity of a pixel in the first image as a multiplicative gain on a learned, symmetric weight between a pixel in the second image and a hidden unit. This creates cubically many parameters, which form a three-dimensional interaction tensor. We describe a low-rank approximation to this interaction tensor that uses a sum of factors, each of which is a three-way outer product. This approximation allows efficient learning of transformations between larger image patches. Since each factor can be viewed as an image filter, the model as a whole learns optimal filter pairs for efficiently representing transformations. We demonstrate the learning of optimal filter pairs from various synthetic and real image sequences. We also show how learning about image transformations allows the model to perform a simple visual analogy task, and we show how a completely unsupervised network trained on transformations perceives multiple motions of transparent dot patterns in the same way as humans.

1 Introduction ---

A restricted Boltzmann machine (RBM) is a simple learning module in which a layer of visible units that represent the observed data is connected to a layer of hidden units that learn to extract features from the data. To simplify the inference and learning procedures, there are no connections within a layer. RBMs have three important computational properties. First, given an observed data vector on the visible units, it is easy to infer the states of the hidden units because the lack of connections between hidden units makes them conditionally independent, given the states of the visible units. Second, it is easy to produce a reconstruction of the data from the hidden states because the lack of connections between visible units makes them conditionally independent, given the states of the hidden units. Third, there is a simple and efficient learning procedure for updating the weights

on the connections. Each weight update is proportional to the difference between the correlation of a visible and a hidden unit when the hidden units are driven by data and by reconstructions (Hinton, 2002).

These three properties are all preserved when the hidden and visible units receive additional, conditioning inputs from previous states of the visible units (Taylor, Hinton, & Roweis, 2007). The conditioning inputs have the effect of dynamically changing the biases of the units, and the weights on these inputs can be learned by using the derivatives for the biases, provided that the previous states of the visible units are not changed when the data are reconstructed.

A much more powerful way of conditioning on previous visible states is to allow them to have multiplicative effects on the weights of the RBM rather than additive effects on the biases. Boltzmann machines that contain multiplicative interactions between more than two units are known in general as higher-order Boltzmann machines (Sejnowski, 1987). Memisevic and Hinton (2007) show how conditioning an RBM using multiplicative interactions among hidden, visible, and conditioning variables leads to a type of higher-order Boltzmann machine that retains the computational benefits of RBMs, like being amenable to contrastive divergence training and allowing for efficient inference schemes that use alternating Gibbs sampling.

Unfortunately, the model suffers from an explosion in the number of parameters, which scales as the product of conditioning, hidden, and visible variables. In particular, when image transformations are modeled, the number of variables in each of these three groups of units is approximately equal to the number of pixels in an image, which restricts the applicability to small image patches. As a partial solution to this problem Memisevic and Hinton (2007) suggest adding a preprocessing layer to the model that can reduce the dimensionality of the input image. However, they note that an extra layer on the output image would not be so easy to accommodate, as it is not compatible with conditional contrastive divergence learning.

One could perform dimensionality reduction on both images in a separate preprocessing step. Unfortunately, this decouples feature extraction from learning, yielding features that are not optimized to represent transformations, only the (separate) static structure in input and output images. In particular, in the absence of any structure within the images (e.g., when observing transformed random images), such a feature extraction scheme would be useless, as there would be no static structure to learn from the images. The fact that there can be regularities in the way the images transform would be entirely ignored by a separately trained feature extraction module.

In this letter, we describe a factorization of the interaction tensor in a higher-order RBM that reduces the number of parameters while retaining the model's ability to learn complicated transformations from real-world images. Our factorization leads to a filter-matching interpretation of the

three-way model in which transformations are modeled with pairs of filters. Feature extraction is an integral part of training the three-way interaction model. As a result, training on transformed images yields pairs of linear filters that are optimized to represent the observed class of transformations. The model discovers Fourier components as the optimal filters for global shifts, for example, or a log-polar variant of the Fourier transform for representing global rotations. We demonstrate the extraction of these and other features using both synthetic and real image sequences. We also show experimentally how the factorization can improve training efficiency, and we demonstrate the use on larger image patches.

The task of learning about image transformations is closely related to the problem of dealing with visual invariances. In fact, knowledge about allowable transformations can be used to build systems that ignore these transformations and thus perform invariant recognition (e.g., Simard, LeCun, Denker, & Victorri, 1992). When transformations are not known a priori, are not stationary, or are difficult to model, we can still perform invariant recognition using a system that can learn about allowable transformations from training examples, as suggested, for example, by Rao and Ruderman (1999), Olshausen, Cadieu, Culpepper, and Warland (2007), Miao and Rao (2007), and Memisevic and Hinton (2007). An early approach to learning image transformations in a biologically plausible way, using temporal coherence, is described in Foldiak (1991). More recently, several multilinear and spatiotemporal filtering models were suggested for this task (e.g., Tenenbaum & Freeman, 2000; Olshausen, 1994; Olshausen et al., 2007; Rao & Ballard, 1997; Rao & Ruderman, 1999; Grimes & Rao, 2005; van Hateren & Ruderman, 1998; Vasilescu & Terzopoulos, 2002). The idea behind these approaches is to use multiple independent groups of hidden variables in order to model the multiple independent latent factors that contribute to the variability in the data. Bilinear models contain exactly two groups of hidden variables: a “what” component accounts for the images and a “where” component for the transformations (Olshausen et al., 2007).

Unfortunately, the presence of two or more groups of hidden variables leads to a chicken-and-egg problem that can make learning and inference difficult (Tenenbaum & Freeman, 2000). Miao and Rao (2007), Olshausen et al. (2007), and Memisevic and Hinton (2007) overcome this issue by conditioning transformed images on untransformed ones rather than trying to simultaneously model both the images and the transformations. (More specifically, Olshausen et al., 2007, discuss both a conditional and a bilinear model. The conditional model is referred to as “remapping model” in their paper.) As a result, the models contain only a single group of hidden variables, and these are responsible only for encoding transformations. Operating on raw pixels rather than features, however, leads to the issues discussed above.

In this letter, we show how factorization allows us to define a conditional model with a single group of hidden variables that nevertheless makes it

possible to extract features from the images and thus to model “what” components in addition to “where” components. The “what” components, however, are tied to the observed transformations. In particular, when the images are random, they turn into filters that represent only the transformations. As we shall show, the reason that there is no chicken-and-egg problem is that the factors, which take on the role of the feature extractors, behave like deterministic functions rather than stochastic hidden variables. Inference and learning are therefore basically the same as in an unfactored model.

We demonstrate the use of the factorized model in various tasks, including an analogy-making problem and the task of representing optical flow in the presence of transparency.

2 Unfactored Higher-Order Restricted Boltzmann Machines ---

One way to learn about image transformations from pairs of images x and y is by modeling the conditional distribution $p(y | x) = \sum_{\mathbf{h}} p(y, \mathbf{h} | x)$ of the transformed output image y given the original input image x , where \mathbf{h} is a vector of hidden variables. We restrict our attention to binary images and hidden variables for now that can take on the values 0 or 1. However, this restriction can easily be removed, as we discuss later.

The idea behind this approach to modeling image transformations is that by training on a set of image pairs (x^α, y^α) , the hidden variables can learn from data how to efficiently encode transformations. In contrast to the well-known task of learning static structure from independent images (e.g., Olshausen & Field, 1996; Bell & Sejnowski, 1997), the goal here is to learn about structure in how images change over time (Memisevic & Hinton, 2007). As in the case of still images, transformations of images in, say, natural video, can be highly structured and regular, which suggests that hidden variables can be used to discover and represent these transformations efficiently.

To be able to capture all possible correlations among input, output, and hidden variables, Memisevic and Hinton (2007) suggest using the following three-way energy function in order to define the conditional distribution:

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \sum_{ijk} w_{ijk} x_i y_j h_k + \sum_k w_k^h h_k + \sum_j w_j^y y_j, \quad (2.1)$$

where i , j , and k index input, output and hidden units, respectively; x_i is the binary state of input pixel i ; y_j is the binary state of output pixel j ; and h_k is the binary state of hidden unit k . The components w_{ijk} of a three-way interaction tensor w connect units x_i , y_j , and h_k . The terms $\sum_k w_k^h h_k$ and $\sum_j w_j^y y_j$ are bias terms used to model the base rates of activity of the hidden and observable units, respectively. In general, a higher-order

Boltzmann machine can also contain higher-order bias terms (Memisevic & Hinton, 2007), but we do not use these in this letter. Since hidden units encode mappings from \mathbf{x} to \mathbf{y} , we refer to them also as mapping units. We use *hidden units* and *mapping units* interchangeably in this letter.

The energy function is turned into a probability distribution by exponentiating and normalizing:

$$p(\mathbf{y}, \mathbf{h} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x})), \quad (2.2)$$

$$Z(\mathbf{x}) = \sum_{\mathbf{h}, \mathbf{y}} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x})). \quad (2.3)$$

Note that normalization is over \mathbf{y} and \mathbf{h} , thus defining the conditional distribution $p(\mathbf{y}, \mathbf{h} | \mathbf{x})$ rather than the joint $p(\mathbf{y}, \mathbf{h}, \mathbf{x})$, which simplifies inference and learning (Memisevic & Hinton, 2007).

Note that the energy function contains multiplicative interactions between triplets of units x_i , y_j , and h_k , but not between any pairs of output units or pairs of hidden units. As a result, the conditionals $p(\mathbf{y} | \mathbf{h}; \mathbf{x})$ $p(\mathbf{h} | \mathbf{y}; \mathbf{x})$ factorize, which, as in RBMs, greatly simplifies both learning and inference, since it facilitates efficient Gibbs sampling. More specifically, we have:

$$p(\mathbf{y} | \mathbf{h}; \mathbf{x}) = \prod_j p(y_j | \mathbf{h}; \mathbf{x}) \quad (2.4)$$

$$\text{with } p(y_j = 1 | \mathbf{h}; \mathbf{x}) = \frac{1}{1 + \exp(-\sum_{i,k} w_{ijk} x_i h_k - w_j^y)} \quad (2.5)$$

and

$$p(\mathbf{h} | \mathbf{y}; \mathbf{x}) = \prod_k p(h_k | \mathbf{y}; \mathbf{x}) \quad (2.6)$$

$$\text{with } p(h_k = 1 | \mathbf{y}; \mathbf{x}) = \frac{1}{1 + \exp(-\sum_{i,j} w_{ijk} x_i y_j - w_k^h)}. \quad (2.7)$$

Figure 1 (left) shows two ways of interpreting the three-way model. Like a mixture of experts, the model can be viewed as a nonlinear blend of linear regressors, where, in contrast to a mixture of experts, the number of mixture components is exponential in the number of hidden units. This allows for a factorial decomposition of the dependencies between \mathbf{x} and \mathbf{y} (left-most plot). Alternatively, the model can be viewed as an RBM whose connections are modulated by input pixels, which can consequently “vote” for linear filters that jointly model the output image (Memisevic & Hinton, 2007).

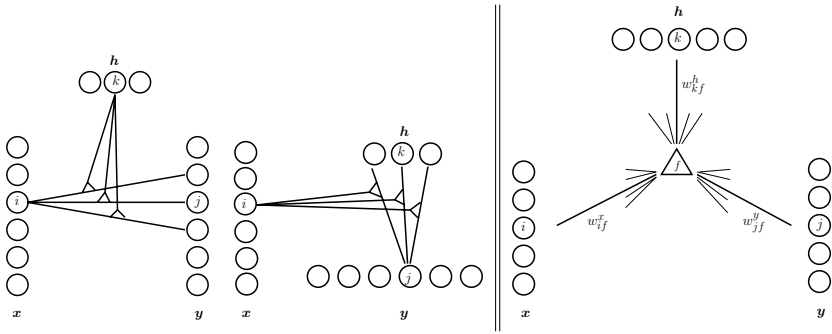


Figure 1: (Left) Two views of an unfactored gated RBM. (Right) One factor in a factored gated RBM.

The model can also be viewed as a standard RBM modeling $p(\mathbf{y}, \mathbf{h})$, but with weights w_{jk} that are defined as linear functions $\sum_i x_i w_{ijk}$ of the inputs \mathbf{x} . Thus, extensions that have been suggested for RBMs typically apply to these models as well. An example is the generalization of the model to use exponential family distributions for hidden and observed variables (Welling, Rosen-Zvi, & Hinton, 2005; Memisevic & Hinton, 2007). Because the energy function is a sum over all input, output, and hidden units (see equation 2.1), the time complexity for inference or one-step inference is $O(IJK)$, where I is the number of input, J the number of output, and K the number of hidden units. In practice, the number of hidden units is typically on the order of the number of input and output units. Thus, when applied to images, learning scales cubically with the number of pixels and is therefore slow even on relatively small image patches. The large number of parameters usually also requires large amounts of training data to generalize well.

3 Factoring Three-Way Interactions

In many applications that require multiplicative interactions, strong underlying regularities allow the data to be modeled using less than the cubically many parameters implied by three-way weights. Image transformations, for example, do not involve arbitrary motions of individual pixels. This suggests that it may be possible to represent the required three-way interaction tensor using far fewer parameters.

The central idea of this letter is to factor the interaction tensor in a particular way. The three-way energy of a joint configuration of the visible and hidden units is defined as

$$-E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = \sum_{f=1}^F \sum_{ijk} x_i y_j h_k w_{if}^x w_{jf}^y w_{kf}^h + \sum_k w_k^h h_k + \sum_j w_j^y y_j, \quad (3.1)$$

where f indexes the factors. In other words, the $I \times J \times K$ parameter tensor is replaced by three matrices with sizes $I \times F$, $J \times F$, and $K \times F$. If the number of factors is comparable to the number of units in the visible and hidden groups, the factorization reduces the number of parameters from $O(N^3)$ to $O(N^2)$. Using this factorization, the weight w_{ijk} in equation 2.1 is implemented by $\sum_f w_{if}^x w_{jf}^y w_{kf}^h$. The bias terms remain unchanged. As in the unfactored model, equation 2.3, exponentiation and normalization define the conditional $p(\mathbf{y}, \mathbf{h} | \mathbf{x})$.

It is instructional to rewrite the energy, using the distributive law, as

$$\begin{aligned}
 -E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = & \sum_f \left(\sum_i x_i w_{if}^x \right) \left(\sum_j y_j w_{jf}^y \right) \left(\sum_k h_k w_{kf}^h \right) \\
 & + \sum_k w_k^h h_k + \sum_j w_j^y y_j,
 \end{aligned} \tag{3.2}$$

which shows that each factor f corresponds to a triplet of linear filters—one for the input, one for the output, and one for the hidden unit activities. Figure 1 (right plot) is a graphical representation of this way of factoring the interaction tensor. The three filter responses for each factor are matched multiplicatively. As a result, each pattern of activity in the hidden units corresponds to one way of matching input filter responses to output filter responses. Learning amounts to finding a suitable set of corresponding filters that can represent the transformations in the training data.

Unlike the common unsupervised approaches to learning image filters, like sparse coding (Olshausen & Field, 1996), ICA (Bell & Sejnowski, 1997), and others, here it is the relationship between images that filters need to account for, not the structure within a single image. As we shall show, the model learns highly structured filters even when trained on images that are entirely random.

3.1 Inference Procedure for RBMs with Multiplicative Factors. Depending on the application, there are several ways to define the inference task for a trained model. One common task is to compute $p(\mathbf{h} | \mathbf{y}, \mathbf{x})$ from a given test image pair (\mathbf{x}, \mathbf{y}) . It amounts to inferring the transformation from an image pair. This task can be solved easily and efficiently because of the factorization of the conditional distribution (see equation 2.6). Another possible task is to compute $p(\mathbf{y} | \mathbf{h}, \mathbf{x})$ from an input image \mathbf{x} and a given, fixed transformation \mathbf{h} . This task can be thought of as a form of analogy making, as we shall discuss in detail in section 4.2. It can be solved easily, too, because of the factorization (see equation 2.4).

A third possible task consists in computing, or sampling from, $p(\mathbf{y} | \mathbf{x})$ after seeing a test input image \mathbf{x} (which was not part of the training data). This task requires marginalizing over the hidden variables \mathbf{h} . The exact

computation is intractable if the number of hidden variables is large. However, since the conditional distribution takes the form of an RBM, we can use alternating Gibbs sampling to sample from it, as is common in RBMs. In particular, the factorization of the conditionals (see equations 2.4 and 2.6) suggests sampling from $p(\mathbf{y}, \mathbf{h} | \mathbf{x})$ by alternating between sampling from $p(\mathbf{y} | \mathbf{h}, \mathbf{x})$ and sampling from $p(\mathbf{h} | \mathbf{y}, \mathbf{x})$.

To perform Gibbs sampling in the factored model, each hidden unit must compute the energy difference between its two states. So the input that a hidden unit, h_k , requires from the factors in order to perform Gibbs sampling by using the logistic function is

$$-\Delta E_k = \sum_f w_{kf}^h \sum_i x_i w_{if}^x \sum_j y_j w_{jf}^y + w_k^h. \quad (3.3)$$

The input that a visible unit in the second image must receive from the factors is

$$-\Delta E_j = \sum_f w_{jf}^y \sum_i x_i w_{if}^x \sum_k y_k w_{kf}^h + w_j^y, \quad (3.4)$$

where i indexes pixels in the first image. This resembles belief propagation because the message $\sum_i x_i w_{if}^x \sum_j y_j w_{jf}^y$ that factor f sends to all the hidden units does not involve the input it receives from the hidden units. In this respect, the deterministic factors act very differently from stochastic “units,” which send the same message (i.e., their state) everywhere. If the factors are replaced by a layer of intermediate stochastic units or mean field units, the hidden units cease to be conditionally independent, so inference and learning become much more difficult.

3.2 Learning Procedure for RBMs with Multiplicative Factors. To train the factored model, we need to maximize the average log probability $L = \sum_\alpha \log p(\mathbf{y}^\alpha | \mathbf{x}^\alpha)$ of a set of training pairs $\{(\mathbf{x}^\alpha, \mathbf{y}^\alpha)\}$. The derivative of the negative log probability with regard to any element θ of the parameter tensor is given by

$$-\frac{\partial L}{\partial \theta} = \sum_\alpha \left\langle \frac{\partial E(\mathbf{y}^\alpha, \mathbf{h}; \mathbf{x}^\alpha)}{\partial \theta} \right\rangle_{\mathbf{h}} - \left\langle \frac{\partial E(\mathbf{y}, \mathbf{h}; \mathbf{x}^\alpha)}{\partial \theta} \right\rangle_{\mathbf{h}, \mathbf{y}}, \quad (3.5)$$

where $\langle \rangle_z$ denotes the average with regard to variable \mathbf{z} . By differentiating equation 3.1 with regard to the model parameters, we get:

$$-\frac{\partial E_f}{\partial w_{kf}^h} = h_k \sum_i x_i w_{if}^x \sum_j y_j w_{jf}^y, \quad (3.6)$$

$$-\frac{\partial E_f}{\partial w_{jf}^y} = y_j \sum_i x_i w_{if}^x \sum_k h_k w_{kf}^h, \quad (3.7)$$

$$-\frac{\partial E_f}{\partial w_{if}^x} = x_i \sum_j y_j w_{jf}^y \sum_k h_k w_{kf}^h, \quad (3.8)$$

where E_f denotes the energy contributed by a single factor f (one summand on the right-hand side of equation 3.2). Derivatives of the bias terms are the same as in a standard RBM: $-\frac{\partial E}{\partial w_k^h} = h_k$ and $-\frac{\partial E}{\partial w_j^y} = y_j$.

Computing the average over all hidden and observable units in equation 3.5 is intractable. But we can approximate the averages using alternating Gibbs sampling, as discussed in section 3.1. In our experiments, we use contrastive divergence learning (Hinton, 2002), which amounts to performing each gradient update by starting the Gibbs sampler at the training observations, \mathbf{y}^t , followed by performing a single Gibbs iteration.

4 Experiments

4.1 Learning Filters That Represent Transformations. Since the factorization leads to a matching of filter responses, an interesting question to ask is what forms the filter pairs take on when trained on different types of transformation.

To answer this question, we trained the factorized model with binary hidden and binary observable units on transformations of random dot images.¹ We experimented with a variety of image sizes, ranging from 8×8 to 40×40 pixels, all with similar results. (Unlike the nonfactorized model, training on patches larger than 12×12 pixels is fairly efficient.) We trained on various kinds of transformation, including shifts, scaling, rotation, affine, and convolution with random kernels, as well as split-screen settings with multiple transformations in one image. We trained the models to predict transformed images from the originals as described in section 3.2. Images were generated randomly with typically about 10% of the pixels “on” and the rest “off.” Each class of transformation furthermore contains parameters (e.g., direction of shift, scaling amount, rotation angle) that were chosen independently and randomly for each example image pair. Some of the transformations, like shifts and scaling, are fairly local in the sense that each position in the image is moved by no more than about three pixels. Others, like rotations in section 4.2, are global and can move pixels across the whole image. We generated the training images on the fly and trained the model online on the resulting stream of image pairs using batches of

¹A Python implementation of the model, along with example code to apply the model on synthetic transformations, is available online at learning.cs.toronto.edu/~rfm/factored/.

size 20 to 100 for each gradient update. We used contrastive divergence learning with a small learning rate (usually about 0.01) in the experiments.

In all the experiments, we used between 50 and 200 hidden units and usually 100 factors. The exact number of hidden units and factors did not have a strong influence on the result. The reason is that the numbers of hidden units and factors are already in a regime where they can reconstruct the transformed images almost perfectly. Since we trained the models without using weight decay or any other kind of regularization, there were no additional parameters of the cost function to be set. The model parameters were initialized to small, random values in all experiments.

Despite the nonlinearity of the objective function, we did not encounter any problems with local optima. In particular, multiple runs with different random initializations and different random instantiations of the training data tend to give qualitatively very similar results (up to indeterminacies, like ordering of the filters). We therefore did not make use of any tricks like random restarts to avoid local optima. Although the robustness of the optimization against local optima in these types of model (as in many neural network models in general) is not well understood theoretically, the use of small parameter initializations and small learning rates is likely to be part of the reason. In particular, small parameter initializations and learning rates can be thought of as implementing an annealing (or “homotopy”) scheme. With small parameter values, the model behaves essentially like a linear, or capacity-constrained, model, whose objective function is not plagued much by local optima; the use of a slow learning rate in turn results in a slow increase in model complexity as learning progresses, which allows the optimization to track the global or a good local optimum.

There is no structure at all in the images themselves, which are composed of random pixels. The only source of structure in the data comes from the way the images transform. Figure 2 depicts the filters learned from various transformations and shows that they are nevertheless highly structured and tuned explicitly to the transformation they were trained on. The plots show the strengths of the connections to the input or output pixels using gray values for each factor (brighter means larger).

When trained on shifts in random directions, the model learns to extract a two-dimensional Fourier basis on both the input and output images. Each output filter is exactly a phase-shifted version of the corresponding, matched input filter (see Figure 2, top). Many of the resulting filter pairs are roughly in quadrature, that is, they show about a 90 degree phase difference. We also found that within each of the two filter banks, many filters tend to come in quadrature pairs.

Random shifts provide a clear illustration of the role of factors and hidden variables. Each factor (given by a pair of corresponding input-output filters in the figure) represents a phase shift for a given frequency-orientation pair, since input and output filter in each pair are tuned to about the same frequency and orientation and differ only in phase. Furthermore, the filters

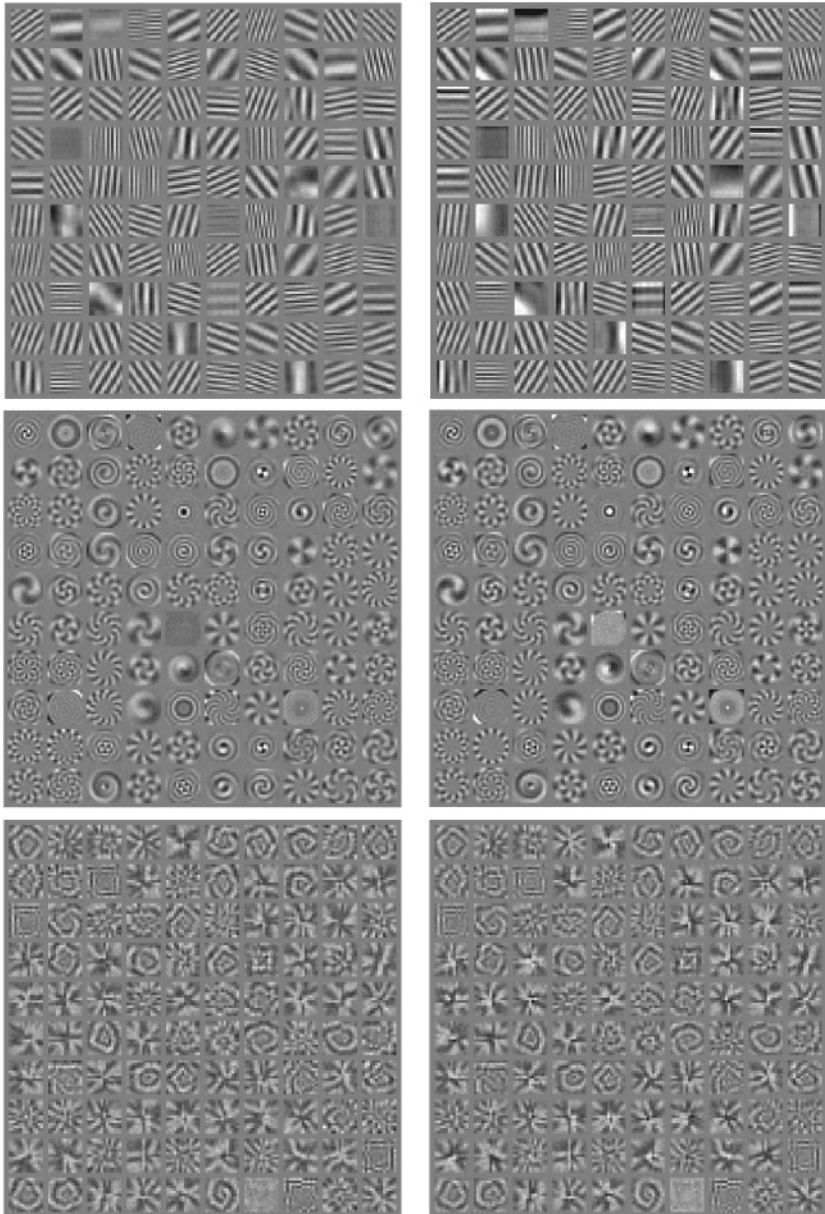


Figure 2: Image filters learned from shifted, rotated, and rescaled random images (top, middle, and bottom row, respectively). There is no structure in the training images, only in how they transform. Left column: input filters w_{if} . Right column: output filters w_{of} .

cover the space of possible frequencies and orientations fairly uniformly. Each hidden unit, on the other hand is connected, with variable strengths, to the factors and thus can detect specific combinations of frequency, orientation and phase shift in the data.

Figure 2 (middle and bottom) shows the input that filters learned from rotations and scalings, respectively. The filters take on very different forms than for shifts. The rotation filters resemble a log-polar version of the Fourier transform (sometimes referred to as “circular harmonics”), with circular and spiral-like features. For scalings the filter set contains mainly concentric shapes that are scaled up or down in the corresponding output filters. We also observed (not shown in these plots) that training on split-screen transformations, where the top half of the image is subjected to a different and independent transformation from the bottom half, leads to filters that are indifferent (zero) in half of the image and contain the usual transformation-specific filter in the other half.

Training on convolutions with random kernels yields Fourier components, as with random shifts.² The filters look very similar to those in Figure 2 (top). The model represents a convolution of an image by first transforming the image into the frequency domain, where it then modifies its spectrum by using a remapping of Fourier components, and subsequently composing the output image from the target components. The model discovers that the frequency domain is useful for representing convolutions entirely by watching randomly convolved images.

Since the learned filter banks are optimized to encode the observed transformations, one should expect the resulting filters within a bank to be relatively invariant with regard to those transformations. To test if this is the case, we used the 100 input filters trained on affine transformations as a linear feature representation for the USPS digit data set (whose images are known to be not very well registered), and obtained a k -nearest-neighbor error rate of 4.4% as compared to 4.8% for the original images and as compared to 11.0% when using 100 PCA features.³

To gain insight into the learning speed of the factored versus the unfactored model, we trained both models on the random shifts for various patch sizes, holding the learning rate constant. Figure 3 shows the projection error for a test set as a function of time and that the speed advantage grows as the patch size increases. There is a slight slowdown in convergence for very small patch sizes, but the gain due to the factorization starts to kick

²In fact, a shift can be thought of as a special case of convolution with a particular kernel, which has the value zero everywhere, except for one pixel, where it has the value one.

³To more rigorously exploit the invariance properties, one should identify the quadrature pairs within a filter bank and use the sum of the squares of the corresponding filter outputs as features. This would provide the phase-invariant magnitude spectrum for the case of Fourier components and an analog for from the more general filters.

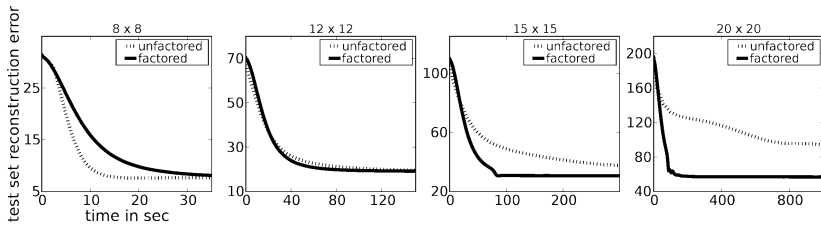


Figure 3: Learning curves gated RBM versus factored-gated RBM for various image sizes.

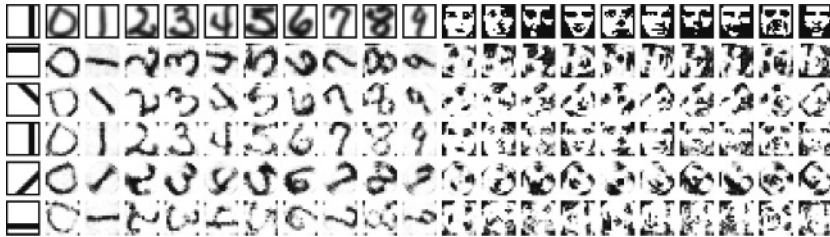


Figure 4: Analogy making. After seeing the transformation of a bar (left-most column), a model trained on rotated pairs of random dot images can apply the same transformation to images that it never saw before, like digits or faces. Prototype images are shown in the top row, and the model's renderings are below each prototype. Neither bars nor digits nor faces were part of the training data.

in at about 12×12 pixels. More important, the plot shows that the gain in performance is a function of input dimensionality.

4.2 Analogy Making. After learning transformations, it is possible to apply the learned transformations by analogy. Presenting a pair of “source” images allows us to extract the binary transformation code h . Holding h fixed and providing a new target image as the input makes it possible to transform the target image the same way as seen in the source image pair (where the definition of “the same” is at the model’s discretion). This is a simple model of analogy making in a restricted domain.

Figure 4 shows digit and face images produced by a model trained on rotated random dots (of size 16×16). We used as the source input a bar image (shown framed, in the top-left corner of the plot) and as the source output one of five rotated versions of the bar (the five images below in the same column). As target inputs, we used digit and face images shown framed in the top row (we picked the first occurrence of each class in the USPS training set and binarized face images of 10 different subjects from the Olivetti faces data set). Below each target image we show the model

output inferred from the corresponding source image pair. The model produces clear, rotated versions of the images. It is important to note that the model never saw any digits or faces during training; the ability to produce digit and face images as the output images is entirely due to generalization.

Analogy making is a capability that arguably relies on the separation of the representation of objects from the representation of how they are related. It is therefore not surprising that the binary encoding of image transformations that the model learns allows it to make analogies in this toy domain. Our approach bears some loose resemblances to the Copycat project introduced by Hofstadter (1984), which is also a stochastic, sampling-based approach to analogy making. One major difference is that in our approach, learning about the domain plays the most important role, after which inference is straightforward, whereas for Copycat, knowledge about the domain, and the representation of objects, is hard-coded, and inference is the main, and most difficult, task. Similarly, Tenenbaum and Freeman (2000) demonstrate an analogy-making approach based on a bilinear model using letter images. They also use a rich, hard-coded representation to restrict the set allowable transformations to ones that are sensible in their domain.

A problem that is related to analogy making is transfer learning, that is, the problem of learning when training data are drawn from a different distribution from the test data. As a practical example, we consider the task of predicting face identity using training faces that are shown in a different orientation from the test faces. We used the binarized Olivetti faces data set, where we rotated the training data with random angles between -90 and 90 degrees. For this task, we subsampled the face images to 32×32 pixels. The data contain 40 subjects with 10 cases each. We used the randomly rotated first five occurrences for each subject as training data and the last five as test data, which amounts to using 200 data points for training and 200 for testing. This is obviously a difficult classification problem (given that there are 40 classes), and the best k NN performance (which occurs at $k = 1$) is 4.5% correct, which is slightly better than random guessing.

After training a model on rotations, we can use it to define a metric that is invariant to rotation by measuring how well the model can reconstruct cases from one another. To determine the (asymmetric) distance between two cases d_1 and d_2 , we first clamp d_1 and d_2 as input and output, respectively, and infer the transformation h . We then apply the inferred transformation by computing a reconstruction \hat{d}_2 for d_2 (where now d_1 and h are clamped). Now we can define the distance between d_1 and d_2 as the reconstruction error $\|d_2 - \hat{d}_2\|^2$ between d_2 and its reconstruction from d_1 . The k NN performance using this metric yields a recognition rate of 36%. Using the model-induced metric effectively allows us to transfer information contained in one data set (faces depicted in one orientation) to a different but related data set (faces depicted in a different orientation).

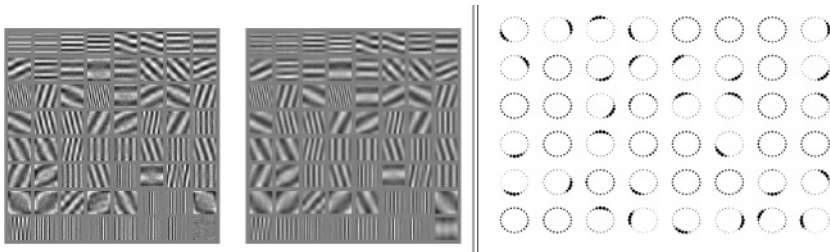


Figure 5: (Left plots) A subset of the filters learned by watching broadcast television. Input filters are on the left and corresponding output filters on the right. (Right plot) Direction selectivity of top-level RBM units after training on shifts of random dot images.

4.3 The Optimal Filter Pairs for Natural Video. To extract filters from natural images, we trained a model with 200 factors and 100 mapping units on image-patch pairs of size 28×28 , cut randomly from adjacent frames of digitized television broadcast. We used the broadcast TV database introduced by van Hateren and Ruderman (1998), which was used also by Memisevic and Hinton (2007) (with much smaller patch sizes). We trained the model on a binarized version of the real-valued data (obtained simply by thresholding). We also experimented with gaussian observables and obtained similar results. We found that the binary model is more stable for large learning rates and therefore much faster to train.

Interestingly, most of the filters we obtain are very clean, phase-shifted Fourier components. The 64 filters that are the largest in terms of Euclidean norm are shown in Figure 5 (many of the 200 factors were almost zero after training and thus do not contribute much to the transformation). This shows that even at a fairly large patch size, uniform shifts of the entire visual field are the predominant mode of variability. However, it is possible that even larger patch sizes, or training on down-sampled versions of the entire frames, could allow us to obtain locally more constrained filters.

If the transformations encountered in the world are responsible in part for the ability to perform invariant recognition, learning about transformations should improve recognition performance. We tested the model's ability to discriminate MNIST digits (yann.lecun.com/exdb/mnist/) by using a metric that is invariant to the transformations seen in the television database. We define the metric as in section 4.2. The best k NN classification error on the data using this metric is 4.6 as compared to 5.1 for Euclidean distance (the optimal k is 5 in both cases). When we shift both the training and test cases by a pixel in random directions, we obtain an even larger difference of 5.1 versus 7.2. Differences of 0.2% classification error are considered statistically significant in this data set, showing that watching TV can improve digit recognition (so it is not a complete waste of time).

5 Extracting Transparent Motion

Memisevic and Hinton (2007) describe how gated RBMs can be used to infer motion from image pairs. We extend that approach and, using the factorized model, show that it is possible to deal with transparency as well. For this purpose, we trained a factored model with 50 hidden units and 200 factors on shifts of random dots (size 13×13). We found that the hidden units tend to encode observed motion in a population code. Therefore, we also trained a second-level RBM with a sparsity penalty using the mapping unit responses as data. The second-level RBM had 50 binary visible units directly connected to 48 binary hidden units.⁴ Figure 5 shows the direction selectivity of the second-level hidden units, which we obtained by presenting random shifts in 24 different directions multiple times and averaging, for each second-level hidden unit, the resulting activities. Most second-level hidden units are either indifferent or clearly tuned to one direction, as shown in the figure.

In order to predict the direction from test-image pairs, we first infer the mapping by computing $p(h | y, x)$, from which we compute the 48 second-level hidden unit activities. Using these activities as weights, we average the orientation tuning fields shown in the right plot of Figure 5 (an approach commonly referred to as vector averaging). Figure 6 depicts the resulting distribution over observed directions when several motions are presented simultaneously. The figure shows that the model uses a multimodal distribution to encode multiple motions and can clearly correctly identify the different directions. We were able to obtain similar results when using the filters learned from broadcast TV (which is not surprising, given the similarities of the filters learned in both settings). However, in the case of real video, several additional difficulties (such as a strong preference for horizontal motion) complicate the results.

It might appear at first sight that there is a much simpler ad hoc approach that could mimic the model's ability to recognize transparent motions. One could independently train or build several motion models, each tuned to a specific direction, and subsequently in some way combine the models' responses to test data showing simultaneous motions. Note, however, that to replicate the described experiment, one would need to build roughly as many models as there are angles that our model can distinguish. Our model's ability to distinguish the different motions relies crucially on its ability to represent several motions using its (binary) code of hidden unit activities (which relies on the fact that it uses a factorial code for representing transformations). In contrast, most common approaches to recognizing motion cannot represent even two transformations.

⁴As an alternative to the RBM, we also experimented with using logistic regression to predict motion direction from mapping unit activities. This improves the prediction accuracy as compared to using the RBM, but it is not an entirely unsupervised approach.

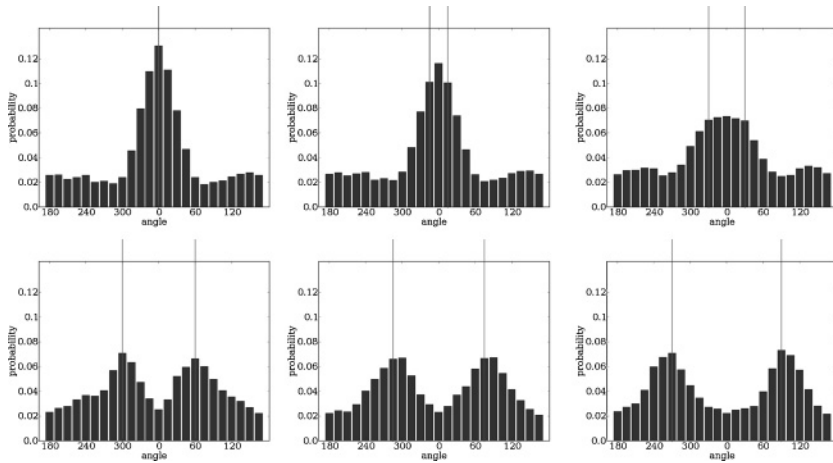


Figure 6: Responses of top-level RBM to two overlaid dot patterns moving in different directions. The two vertical lines represent the actual directions of the two dot patterns; the overlaid bars diagram the resulting probability distribution over directions as perceived by the model.

The model's perception of motion shows several properties that are consistent with findings on human motion perception. The model shows the merging of modes at small angles, which results in averaging of presented motion directions. Interestingly, we found also that at angles slightly larger than those where merging occurs, the model slightly but consistently overestimates the observed angles, which is consistent with the well-known phenomenon of motion repulsion. None of the observed effects were part of the design consideration of the transformation model, whose only goal is to model a conditional probability over output images given an input image.

6 Discussion

In this letter, we introduced a factored version of a conditional higher-order RBM and showed how the model allows us to obtain image filter pairs that are optimal at representing transformations of images. This is in contrast to the well-known work on learning static structure from independent images, and the filters that the model learns are quite different from those typically obtained from static images.

We focused on relatively simple transformations in this letter, like global rotations, translations, and affine transformations. Memisevic and Hinton (2007) and Memisevic (2008) describe experiments showing that the unfactored model is capable of learning more general transformations, such as split-screen transformations (i.e., simultaneous, independent

transformations in two different parts of an image), superresolution, or canvas-like renderings of a photo. Some early experiments with the factored model suggest that such more general transformations can be learned with the factored model too. In particular, we found that training the factored model on split-screen data yields filters that specialize to specific areas of an image, similar to the mapping units in the unfactored model (Memisevic, 2008). A more detailed analysis of the factored model's ability to learn those general transformations as well as potential differences between the factored and unfactored model in terms of the types of transformation they can represent will be the subject of future research.

Besides giving rise to the filter pairs, the factorization can also help reduce the number of model parameters and speed up learning, as we discussed in detail. A possible question to ask is whether one can apply similar kinds of factorization to other graphical models as well. It is important to note in this context that training in our case relies crucially on the particular independence assumptions that make inference fast and simple. In particular, there are no within-group connections—in other words, no connections between mapping units or between output units. The factorization has the useful property that it keeps this independence structure intact and thus does not interfere with inference and learning. It is not immediately obvious to what degree graphical models with entirely different kinds of independence assumptions could benefit from similar factorizations. Undirected models that have a similar independence structure as the model discussed here, such as the various exponential family variants of RBMs, as well as their extensions, however, certainly can.

We used exclusively contrastive divergence learning to train all models in this letter, mainly because it is fast and robust and worked very well in all the experiments. Similar results could probably be achieved using full maximum likelihood learning of the models by using prolonged Gibbs sampling. However, it seems unlikely that there are benefits that would warrant the much longer training time. It is also possible to define nonprobabilistic variations of the model. Memisevic (2008), for example, discusses various nonprobabilistic formulations of the unfactored version of the model, and similar formulations would be possible for the factored model. The nonprobabilistic models can be trained using gradient-based optimization.

We considered only first-order bias terms in this letter, so the only connections we used are factored three-way connections and the (unfactored) simple bias terms. Learned filters therefore need to represent only the transformations. It might be interesting to investigate adding factored second-order biases connecting hidden units to output units, which would separately model, and possibly factor out, purely static structure in the images.

Memisevic and Hinton (2007) describe a convolutional version of the unfactored model, which can be used to model sets of large images with possibly variable sizes. A convolutional version of the factored model is

an interesting direction for further research. Another research direction is the use of sparsity. Imposing sparsity constraints on hidden variables has been a common way of obtaining localized filters in static image coding models (e.g., Olshausen & Field, 1996) and recently also in bilinear models of transformations (e.g., Grimes & Rao, 2005). Sparsity constraints on the mapping units could be useful also in our model and might give rise to more localized (less distributed) codes for the observed motions.

References

- Bell, A. J., & Sejnowski, T. J. (1997). The independent components of natural scenes are edge filters. *Vision Research*, 37, 3327–3338.
- Foldiak, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3(2), 194–200.
- Grimes, D. B., & Rao, R. P. N. (2005). Bilinear sparse coding for invariant vision. *Neural Computation*, 17(1), 47–73.
- Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8), 1771–1800.
- Hofstadter, D. R. (1984). *The copycat project: An experiment in nondeterminism and creative analogies* (A.I. Memo 755). Cambridge, MA: Massachusetts Institute of Technology, 755.
- Memisevic, R. (2008). *Non-linear latent factor models for revealing structure in high-dimensional data*. Unpublished doctoral dissertation, University of Toronto.
- Memisevic, R., & Hinton, G. (2007). Unsupervised learning of image transformations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE Press.
- Miao, X., & Rao, R. P. N. (2007). Learning the Lie groups of visual invariance. *Neural Computation*, 19(10), 2665–2693.
- Olshausen, B. (1994). *Neural routing circuits for forming invariant representations of visual objects*. Unpublished doctoral dissertation, California Institute of Technology.
- Olshausen, B. A., Cadiou, C., Culpepper, J., & Warland, D. K. (2007). Bilinear models of natural images. In *SPIE Proceedings: Human Vision Electronic Imaging XII*. Bellingham, WA: SPIE.
- Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583), 607–609.
- Rao, R. P., & Ballard, D. H. (1997). *Efficient encoding of natural time varying images produces oriented space-time receptive fields* (Tech. Rep.). Rochester, NY: University of Rochester.
- Rao, R. P., & Ruderman, D. L. (1999). Learning Lie groups for invariant visual perception. In M. S. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems*, 11. Cambridge, MA: MIT Press.
- Sejnowski, T. J. (1987). Higher-order Boltzmann machines. In *AIP Conference Proceedings 151 on Neural Networks for Computing* (pp. 398–403). Woodbury, NY: American Institute of Physics.

- Simard, P., LeCun, Y., Denker, J., & Victorri, B. (1992). An efficient algorithm for learning invariances in adaptive classifiers. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition. Conference B: Pattern Recognition Methodology and Systems*. N.p.: IAPR.
- Taylor, G. W., Hinton, G. E., & Roweis, S. T. (2007). Modeling human motion using binary latent variables. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems*, 19. Cambridge, MA: MIT Press.
- Tenenbaum, J. B., & Freeman, W. T. (2000). Separating style and content with bilinear models. *Neural Computation*, 12(6), 1247–1283.
- van Hateren, L., & Ruderman, J. (1998). Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 265(1412), 2315–2320.
- Vasilescu, M. A. O., & Terzopoulos, D. (2002). Multilinear analysis of image ensembles: Tensorfaces. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I* (pp. 447–460). Berlin: Springer-Verlag.
- Welling, M., Rosen-Zvi, M., & Hinton, G. (2005). Exponential family harmoniums with an application to information retrieval. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*, 17. Cambridge, MA: MIT Press.