Preprint: To appear, Neural Computation, 2007

Visual Recognition and Inference Using Dynamic Overcomplete Sparse Learning

Joseph F. Murray
Massachusetts Institute of Technology
Brain and Cognitive Sciences Department
77 Massachusetts Ave. 46-5065
Cambridge, MA 02139
and
Kenneth Kreutz-Delgado
University of California, San Diego
Electrical and Computer Engineering Department
9500 Gilman Dr. Dept 0407
La Jolla, Ca 92093-0407
Email: murrayjf@mit.edu, kreutz@ece.ucsd.edu[*]

April 8, 2007

Abstract

We present a hierarchical architecture and learning algorithm for visual recognition and other visual inference tasks such as imagination, reconstruction of occluded images, and expectation-driven segmentation. Using properties of biological vision for guidance, we posit a stochastic generative world model and from it develop a simplified world model (SWM) based on a tractable variational approximation that is designed to enforce sparse coding. Recent developments in computational methods for learning overcomplete representations (Lewicki and Sejnowski, 2000; Teh et al., 2003) suggest that overcompleteness can be useful for visual tasks, and we use an overcomplete dictionary learning algorithm (Kreutz-Delgado et al., 2003) as a preprocessing stage to produce accurate, sparse codings of images.

Inference is performed by constructing a dynamic multilayer network with feedforward, feedback and lateral connections, which is trained to approximate the SWM. Learning is done with a variant of the backpropagation-through-time algorithm, which encourages convergence to desired states within a fixed number of iterations. Vision tasks require large networks and, to make learning efficient, we take advantage of the sparsity of each layer to update only a small subset of elements in a large weight matrix at each iteration. Experiments on a set of rotated objects demonstrate various types of visual inference, and show that increasing the degree of

[*]Webpages: J.F. Murray: www.jfmurray.org, K. Kreutz-Delgado: dsp.ucsd.edu/~kreutz

overcompleteness improves recognition performance in difficult scenes with occluded objects in clutter.

1 Introduction

Vision, whether in the brain or computer, can be characterized as the process of inferring certain unknown quantities using an input image and predictions or expectations based on prior exposure to the environment. Visual inference includes tasks such as recognizing objects, reconstructing missing or occluded features, imagining previously learned or entirely novel objects, and segmentation (finding which features in a cluttered image correspond to a particular object). Performing these inference tasks requires combining information about the current image (bottom-up processing) and abstract concepts of objects (top-down processing). These tasks can naturally be placed into the framework of Bayesian probabilistic models, and determining the structure and priors for such models is a great challenge both for understanding vision in the brain and for application-oriented computer vision. A primary goal of this paper is to derive an effective probabilistic model of visual inference consistent with current understanding of biological vision.

A number of important properties have emerged from neuroscience: 1) Vision in the brain is a *hierarchical* process with information flowing from the retina to the lateral geniculate nucleus (LGN), occipital and temporal regions of the cortex (Kandel et al., 2000). 2) This hierarchy has extensive *recurrence* with reciprocal connections between most regions (Felleman and Van Essen, 1991). 3) There is also extensive recurrence within cortical regions, as typified by *lateral inhibition* which is a mechanism for how sparse coding can arise (Callaway, 2004). 4) The primary visual cortex (V1) is strikingly *overcomplete*, meaning there are many more cells than are needed to represent the retinal information. In humans, there are over 200-300 V1 neurons per each LGN neuron, and a lesser degree of overcompleteness in other primates (Stevens, 2001; Ejima et al., 2003). 5) The firing patterns of cortical neurons gives evidence for *sparse distributed representations*, in which only a few neurons are active out of a large population, and that information is encoded in these ensembles (Vinje and Gallant, 2000; Quiroga et al., 2005). 6) Finally, even though there are differences between various areas, the basic structure of the cortex is qualitatively similar, and the notion of *cortical similarity* states that the underlying cortical operation should be similar from area to area (Mountcastle, 1978; Hawkins and Blakeslee, 2004). Since these six properties are present in animals with high visual acuity, it is reasonable to assume they are important for inference, and we will adopt them in a network model.

While many computational models of vision have been developed which incorporate some of the above-listed properties (Fukushima and Miyake, 1982; Rao and Ballard, 1997; Riesenhuber and Poggio, 1999; Rolls and Milward, 2000; Lee and Mumford, 2003; Fukushima, 2005), we propose a model which takes into account all six properties. For example, the recognition models of Rolls and Milward (2000) and Riesenhuber and Poggio (1999) do not use feedback (and so are incapable of inference tasks such as reconstruction or imagination), and the dynamic system of Rao and Ballard (1997) does not use overcomplete representations. The use of *learned* overcomplete representations for preprocessing is a new and largely unexplored approach for visual recognition and inference algorithms. Recent developments in learning overcomplete dictionaries (Lewicki and Sejnowski, 2000; Kreutz-Delgado et al., 2003; Teh et al., 2003) and the associated methods for sparse image coding (Murray and Kreutz-Delgado, 2006) now make possible the investigation of their utility for visual inference.

Real world images are high-dimensional data that can be explained in terms of a much smaller number of causes, such as objects and textures. Each object, in turn, can appear in many different orientations but in fact is seen in only one particular orientation. For each orientation, an object can be represented with a concise set features, such as lines, arcs and textures. The key feature of these various types of image descriptions is that they can be represented as *sparse vectors*, where only a few of the many possibly choices suffice to explain the scene. While pixel values of images have non-sparse distributions (they are unlikely to be zero), these more abstract representations are very sparse (each component is likely to be zero), and only a few non-zero components at a time succinctly describe the scene. This intuition, along with the biological evidence for sparsity, is the justification for our use of sparse prior distributions. Other advantages of sparsity include reduced metabolic cost and increased storage capacity in associative memories (Olshausen and Field, 1997).

1.1 Overview and Organization

Beginning with a hypothetical hierarchical *generative world model* (GWM) that is presumed to create images of objects seen in the world, we discuss how the GWM can be used for visual inference (Section 2). The GWM requires the selection of a probability distribution, and a suitable choice is required to create practical algorithms. As a first move, we consider a Boltzmann-like distribution which captures the desired top-down, bottom-up and lateral influences between and within layers, but it is computationally intractable. Then, a *simplified world model* (SWM) distribution is created based on a variational approximation to the Boltzmann-like distribution, and which is specifically designed to model sparse densities (Section 2.4).

By designing a dynamic network that rapidly converges to a self-consistency condition of the SWM, we can perform inference tasks if we have the weights that parameterize the network (Section 3). The dynamic network arises as a way of estimating the fixed-point state of the SWM. Although we consider only the problem of estimating static world-models, generalization to dynamic worlds is also possible. To determine the unknown weights, we develop a learning algorithm based on the backpropagation-through-time algorithm (Williams and Peng, 1990) which operates on the pre-activation state and includes a sparsity-enforcing prior (Section 4). This algorithm can be seen as a natural extension of the sparse-coding principles that are useful in modeling V1 response properties (Olshausen and Field, 1997) to the full visual inference task.

We demonstrate experimentally several types of visual inference including recognition, reconstruction, segmentation and imagination. These simulations show that overcomplete representations can provide better recognition performance than complete codes when used in the early stages of vision (Section 6). A discussion of the biological motivations and comparison to prior work is given in Section 7, and conclusions are drawn in Section 8.

1.2 Notation

$\mathbf{a}$	Activation function parameters
B	Error-related term in learning algorithm

$\mathbf{c}^{(m)}$	Object code for object m, (sparse binary code)
D	Sparsity-enforcing term in learning algorithm
$f(\cdot)$	Sigmoid activation function
$\mathbf{I}\{\cdot\}$	Indicator function, 1 if expression is true, 0 otherwise
J_{PA}	Cost function on pre-activation state, minimized by learning rule
K	Number of images in training set $\mathbb{Y}$
$\mathbf{L}_l$	Lateral weights between units in layer l
M	Number of unique objects in training set
n	Number of layers in network
N	Number of elements in state vector X
$\mathbf{r}$	Number of non-zero elements $\mathbf{r} = [r_1, \ldots, r_n]$, where r_l is the number of non-zero elements in layer l (diversity $\times n$)
$\mathbf{s}$	Size of layers, $\mathbf{s} = [s_1, \ldots, s_n]$, where s_l is the size of layer l
U_t	Network input at time t
$v, \mathbf{v}$	Unit weight sum v (for entire layer $\mathbf{v}_l$), pre-activation function
V	Pre-activation state of all layers
$\widehat{V}_t$	Certainty-equivalence approximation of pre-activation values
$\mathbf{W}_{lm}$	Weights from layer m to layer l
$\mathbb{W}$	Complete weight matrix for all layers (including all $\mathbf{W}_{lm}$ and $\mathbf{L}_l$), $\mathbb{W} \in \mathbb{R}^{N \times N}$
$\mathbf{x}_l$	Activation vector at layer l, expected values of $P(\mathbf{z}_l \mid \mathbf{z}_{l-1}, \mathbf{z}_{l+1})$
X	State vector of all layers, $X = [\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T]^T$
$\check{Y}$	Training data, $\check{Y} = [\check{\mathbf{y}}_1^T, 0, \ldots, 0, \check{\mathbf{y}}_n^T]^T$, where $\check{\mathbf{y}}_1$ is a sparsely-coded image and $\check{\mathbf{y}}_n$ is an object code
Y_t	Dynamic network output at time t
$\mathbb{Y}, \mathbb{V}, \mathbb{U}$	Sets of multiple state vectors Y, V, e.g. $\mathbb{Y} = \{Y^{(1)}, \ldots, Y^{(K)}\}$
$\mathbf{z}_l$	True state of generative model at layer l, binary random vector $\in \{0,1\}^{s_l}$
Z	True state of generative model, all layers, $Z = [\mathbf{z}_1^T, \ldots, \mathbf{z}_n^T]^T$, binary random vector $\in \{0,1\}^N$
$\boldsymbol{\beta}_t$	Indicator vector of whether target values are available for each element of V_t
ε	Error between variational approximation and true state
$\widehat{\varepsilon}$	Error between data set and network approximation $\widehat{V}_t$
ζ	Normalization constant (partition function)
η	Learning rate
λ	Regularization parameter
μ	Target mean for hidden layers
Φ	Error between true and approximate state, $\Phi = Z - X = [\boldsymbol{\phi}_1^T, \ldots, \boldsymbol{\phi}_n^T]^T$
ξ	Energy-like function
τ	Number of time steps network is run for (maximum value of t)
GWM	Generative world model (Boltzmann-like distribution)
NLCP	Neighboring-layer conditional probability

SWM	Simplified world model, variational approx. to Boltzmann-like distribution
DN	Dynamic network that settles to the self-consistency condition of the SWM

2 Generative Models for Visual Inference

In this section, we postulate a hierarchical generative visual-world model (GWM) and discuss its properties, particularly that of independence of one layer given its immediately neighboring layers. We then discuss how the GWM can be used for visual inference tasks such as recognition, imagination, reconstruction, and expectation-driven segmentation. Specific forms of the probability distribution in the model must be chosen, and as a starting point we posit a Boltzmann-like distribution. Since inference with the Boltzmann-like distribution is generally intractable, a variational approximation is developed leading to a simplified world model (SWM). The key assumption of sparsely-distributed activations (prior distributions) is enforced and used extensively. In this section we consider static world-models; in Section 3.1 we will use dynamic networks to implement inference by settling to the fixed points of the SWM.

2.1 Hierarchical Generative Visual-World Model

Images of objects seen in the world can be thought of as being created by a hierarchical, stochastic generative model (the *generative world model*, GWM). While it cannot be rigorously claimed that the real world uses such a model to generate images, the idea of the GWM is a useful fiction that guides the development of learning algorithms (Hinton and Ghahramani, 1997).

For the GWM, we assume a hierarchical binary-state model of the form shown in Figure 1. The number of layers is somewhat arbitrary, though there should be enough layers to capture the structure of the data to be modeled, and four to five appears to be reasonable for images of objects (Riesenhuber and Poggio, 1999; Lee and Mumford, 2003; Hinton et al., 2006). The arrows in Figure 1 indicate that each layer, given the layer directly above it, is independent of higher layers. At the highest level, the vector $\mathbf{z}_5$ is a sparse binary coding of the object in the image, and its value is drawn from the prior distribution $P(\mathbf{z}_5)$. The representation of the particular orientation $\mathbf{z}_4$ of an object depends only on the object representation $\mathbf{z}_5$. The invariant, composite and local features, $\mathbf{z}_3, \mathbf{z}_2$ and $\mathbf{z}_1$, likewise depend only on the layer immediately above them, e.g. $P(\mathbf{z}_3|\mathbf{z}_4, \mathbf{z}_5) = P(\mathbf{z}_3|\mathbf{z}_4)$, and the local features $\mathbf{z}_1$ model the image I. The sequence can be summarized,

$$\mathbf{z}_5 \xrightarrow{P(\mathbf{z}_4|\mathbf{z}_5)} \mathbf{z}_4 \xrightarrow{P(\mathbf{z}_3|\mathbf{z}_4)} \mathbf{z}_3 \xrightarrow{P(\mathbf{z}_2|\mathbf{z}_3)} \mathbf{z}_2 \xrightarrow{P(\mathbf{z}_1|\mathbf{z}_2)} \mathbf{z}_1 \xrightarrow{P(I|\mathbf{z}_1)} I \ . \tag{2.1}$$

The joint distribution of the image and generative states $\mathbf{z}_l$ is,

$$P(I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) = P(I|\mathbf{z}_1)P(\mathbf{z}_1|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_4)P(\mathbf{z}_4|\mathbf{z}_5)P(\mathbf{z}_5) \ , \tag{2.2}$$

where each layer $\mathbf{z}_l$ is a binary vector of size s_l.

We postulate that the $\mathbf{z}_l$ are *sparse*, i.e., they have very few non-zero components (Olshausen and Field, 1997). For example, in every image only a few of all possible objects will be present, and each object will only be in one of its possible orientations, and so forth. Sparsity is proportional to the number of zero

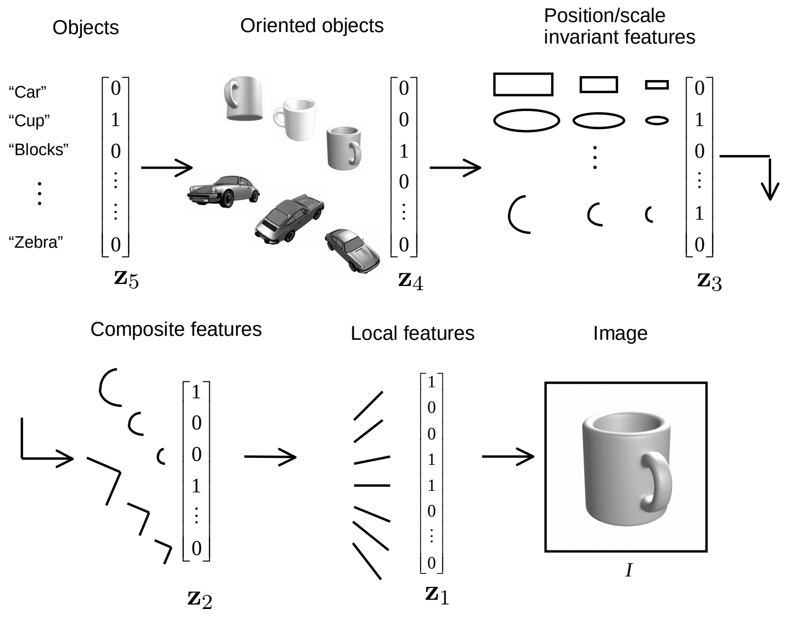


Figure 1: Hierarchical generative visual-world model (GWM) for objects. At each layer $\mathbf{z}_l$, the image can be represented by a large (possibly overcomplete) sparse vector. In this generative model, each layer is a binary random vector which, given only the layer immediately above it in the hierarchy, is independent of other higher layers.

components in a vector $\mathbf{z} \in \mathbb{R}^n$, $sparsity \equiv \#\{z_i = 0\}/n$. A related quantity, *diversity*, is proportional to the number of non-zero components, $diversity \equiv \#\{z_i \neq 0\}/n = 1 - sparsity$. Many studies have confirmed that natural images can be represented accurately by sparse vectors, corresponding to $\mathbf{z}_1$ (Olshausen and Field, 1996; Kreutz-Delgado et al., 2003; Murray and Kreutz-Delgado, 2006). These studies have mainly dealt with small patches of images (on the order of 8x8 to 16x16 pixels), and it is clear that features larger than such patches will be represented non-optimally. This further redundancy in larger-scale features can be reduced at higher levels, which can also have the property of sparseness.

Neighboring Layer Conditional Probability (NLCP). For a middle layer $\mathbf{z}_l$ given all the other layers, we find that $\mathbf{z}_l$ conditioned on its immediate neighbors $\mathbf{z}_{l-1}, \mathbf{z}_{l+1}$ is independent of all the remaining layers. For example,

$$\begin{aligned} P(\mathbf{z}_2|I, \mathbf{z}_1, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) &= \frac{P(I|\mathbf{z}_1)P(\mathbf{z}_1|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_4)P(\mathbf{z}_4|\mathbf{z}_5)P(\mathbf{z}_5)}{P(I|\mathbf{z}_1)P(\mathbf{z}_1|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_4)P(\mathbf{z}_4|\mathbf{z}_5)P(\mathbf{z}_5)} \\ &= \frac{P(\mathbf{z}_1|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_3)}{P(\mathbf{z}_1|\mathbf{z}_3)} \; . \end{aligned} \tag{2.3}$$

For an arbitrary layer we can find the *neighboring layer conditional probability* (NLCP),

$$P(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}) = \frac{P(\mathbf{z}_{l-1}|\mathbf{z}_l)P(\mathbf{z}_l|\mathbf{z}_{l+1})}{P(\mathbf{z}_{l-1}|\mathbf{z}_{l+1})} \qquad \text{(NLCP)} \; . \tag{2.4}$$

This important independence assumption is equivalent to saying that each layer learns about the world only through its neighboring layers (Lee and Mumford, 2003).[1] Returning to the joint distribution and substituting in the NLCPs,

$$\begin{aligned} P(I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) &= P(I|\mathbf{z}_1) \cdot P(\mathbf{z}_1|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_3) \cdot P(\mathbf{z}_3|\mathbf{z}_4)P(\mathbf{z}_4|\mathbf{z}_5) \cdot P(\mathbf{z}_5) \\ &= P(I|\mathbf{z}_1) \cdot P(\mathbf{z}_2|\mathbf{z}_1, \mathbf{z}_3)P(\mathbf{z}_1|\mathbf{z}_3) \cdot P(\mathbf{z}_4|\mathbf{z}_3, \mathbf{z}_5)P(\mathbf{z}_3|\mathbf{z}_5) \cdot P(\mathbf{z}_5) \end{aligned} \tag{2.5}$$

So, the joint can be recovered given the NLCP and additional terms. Of course, other factorizations of the joint are possible, but these are also consistent with the NLCP for their respective layers (Brook, 1964). [2]

Properties of Generative World Model (GWM). We now summarize the four properties of our generative world model (GWM). (1) There is a hierarchy of n hidden-layer vectors $\mathbf{z}_1, \ldots, \mathbf{z}_n$ that model each image I. (2) Each layer is independent of all higher layers given the neighboring layer above, $P(\mathbf{z}_l|\mathbf{z}_{l+1}, \ldots, \mathbf{z}_n) = P(\mathbf{z}_l|\mathbf{z}_{l+1})$. (3) Each layer is independent of all lower layers given the neighboring layer below, $P(\mathbf{z}_l| \mathbf{z}_{l-1}, \ldots, \mathbf{z}_1) = P(\mathbf{z}_l|\mathbf{z}_{l-1})$ (as shown in Murray, 2005, pg. 28). (4) Given its immediate neighboring layers, a layer $\mathbf{z}_l$ is independent of all other higher and lower layers, $P(\mathbf{z}_l|I, \mathbf{z}_1, \ldots, \mathbf{z}_n) = P(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$.

2.2 Types of Inference: Recognition, Imagination, Reconstruction and Expectation-Driven Segmentation

For object recognition, the goal is to infer the highest layer representation $\mathbf{z}_n$ given an image I. However, recognition is only one type of inference that might be required. Another type is running a model generatively using a high-level object representation to *imagine* an image of that object. In the brain, imagining a particular instance of an object will not correspond to the level of detail in the retinal representation, but there is evidence of activity in many of the lower visual areas (such as medial temporal, V1 and V2) during imagination (Kosslyn et al., 1997).

Certain types of inference involve the use of top-down influences interacting with bottom-up inputs. For example, given a partially occluded image that has been recognized by higher layers, top-down influences can be used to *reconstruct* the hidden parts of the object (i.e. those features that are most likely given the input). Another type of inference is *expectation-driven segmentation*, where a prediction is presented at a higher level which may be used to explain cluttered, incomplete or conflicting inputs at the lowest layer, and the desired output is the segmented object at the first layer (Grossberg, 1976; Rao and Ballard, 1997; Hecht-Nielsen, 1998). The expectation input (higher-layer, top-down) must come from a source external to the visual system, which in the brain could be higher cortical areas or other senses, and in computer vision could be dependent on the task or provided by a user. If we wish to find which objects are in a cluttered

[1]The NLCP is closely related to the *Markov blanket*, which is defined for a single node in a Bayesian network as that node's parents, children and children's parents. The NLCP is defined over all the units in a given layer.

[2]Brook (1964) proves that any system specified by the NLCP, $P(z_j|z_i, i \neq j) = P(z_j|z_{j-1}, z_{j+1})$, has a joint distribution that can be factored as $P(z_1, \ldots, z_n) = \Pi_{i=1}^{n+1} Q_i(z_i, z_{i-1})$, which is the joint factorization of a simple Markov chain. This proof is for the case of scalar z, but since our $\mathbf{z}_l$ are binary vectors, they can be equivalently represented as scalar integer variables $\in \{1 \ldots 2^{s_l}\}$. Thus, any system defined by the vector NLCP is consistent with a joint distribution that can be specified as the product of neighboring-layer factors, i.e. the Markov assumption in eq. (2.2).

Table 1: Types of inference that can be performed with the hierarchical generative world model (GWM) and the types of information flow required (bottom-up or top-down). We wish to find a good approximation to the layer $\mathbf{z}_l$ of interest. The approximation used is the expected value of $\mathbf{z}_l$ under the variational approximation, $E_Q[\mathbf{z}_l] = \mathbf{x}_l$ as discussed in Section 2.4.

			Requires	
Type of Inference	Inputs	Outputs	Bottom-up	Top-down
Recognition	$(I \rightarrow \mathbf{z}_1)$	$\mathbf{z}_n$	Y	N
Imagination	$\mathbf{z}_n$	$(\mathbf{z}_1 \rightarrow I)$	N	Y
Reconstruction	$(I \rightarrow \mathbf{z}_1)$	$(\mathbf{z}_1 \rightarrow I)$	Y	Y
Exp.-driven seg.	$(I \rightarrow \mathbf{z}_1)$, $\mathbf{z}_n$	$(\mathbf{z}_1 \rightarrow I)$	Y	Y
Exp.-driven det.	$(I \rightarrow \mathbf{z}_1)$, $\mathbf{z}_n$	$\mathbf{z}_n$	Y	Y

scene (i.e., the desired output is the highest-layer object representation) based on prior knowledge of what might be there (higher-layer input), we perform *expectation-driven detection*. If the high-level prediction about the scene is consistent with the input, the system converges with the expectation at the highest layer and the prediction is confirmed. If the system converges to a different pattern, this indicates that the expected object is not present (which could be considered a state of surprise). Table 1 shows types of inference and the necessary information flow (top-down or bottom-up) needed in the model. As discussed below, we use a sparse-coding algorithm to transform the image into the first layer representation, $\mathbf{z}_1$, and vice versa (denoted by $\rightarrow$ in the table).

2.3 Boltzmann-like Distributions for Layer-Conditional Probabilities

Our next task is to postulate a form for the GWM distributions P that is powerful enough to generate the images seen in the world. A common choice in probabilistic modeling is the *Boltzmann distribution*, $P(\mathbf{z}) = \zeta^{-1} \exp(-\beta\xi(\mathbf{z}))$, where the probabilities are related to a function ξ that assigns an energy to each state, ζ is a normalizing function, and β is a constant (which is a degree-of-randomness parameter related to temperature in physical systems, $\beta \propto T^{-1}$) (Hopfield, 1982; Hinton and Sejnowski, 1983; Hertz et al., 1991). In thermodynamics and physical systems such as magnetic materials, the energy function captures the influence of each particle on its neighbors, where lower-energy states are more probable. The energy function usually has the form $\xi(\mathbf{z}) = -\frac{1}{2}\sum_{ij} w_{ij} z_i z_j$, where w_{ij} is the symmetric interaction weight ($w_{ij} = w_{ji}$) between z_i and z_j. In the context of associative memories, the weights of the energy function are adjusted so that learned patterns form low-energy basins of attraction (e.g. using the Boltzmann machine learning rule, Ackley et al., 1985).

The Boltzmann distribution requires the weights w_{ij} to be symmetric and have zero self-energy $w_{ii} = 0$ (Kappen and Spanjers, 2000). Three main advantages of symmetric weights are: (1) A dynamic network with symmetric interactions is guaranteed to be asymptotically stable and settle to a fixed point (the "zero-temperature" solution) which minimizes the Boltzmann energy function (Mezard et al., 1987). (2) There is a procedure (Gibbs sampling with simulated annealing) which generates samples from this distribution at a given non-zero temperature T. (3) Given a gradual enough annealing schedule for reducing T, Gibbs sampling will track the global minimum-energy state (highest probability state) of the network and guarantee convergence to the zero-temperature solution as the temperature is lowered (Geman and Geman, 1984).

While the above properties are attractive, and help explain the wide interest in the Boltzmann distribution and the Boltzmann machine, they may be of limited use in practice. It often takes considerable time for a stochastic network with symmetric weights to settle to an equilibrium state, possibly longer than a brain or artificial network has to make a decision (Welling and Teh, 2003), which accounts for the interest in simplifying approximations such as mean-field annealing (Peterson and Anderson, 1987). Furthermore, it has also been argued that the use of asymmetric weights can improve performance (such as by suppressing spurious memory states) and has greater biological plausibility (Parisi, 1986; Crisanti and Sompolinsky, 1988; Sompolinsky, 1988; Gutfreund, 1990; Apolloni et al., 1991; Kappen and Spanjers, 2000; Chengxiang et al., 2000). An additional motivation for admitting asymmetric weights is the notion that in hierarchical networks designed for invariant recognition, the relative strengths of the feedforward and feedback pathways will need to be different. Since neurons in higher layers will tend to require inputs from multiple units to activate, the relative strength of the feedback connections to those units must be stronger than the feedforward weights to enable lower layer activity (i.e., generative ability). The primary deterrent to the use of asymmetric weights is the difficulty associated with ensuring asymptotic stability of the resulting algorithms, which involves the use of significantly more complex stability arguments (Apolloni et al., 1991).

We allow for asymmetric weights and sidestep the stability issue by working within a finite-time horizon framework. The resulting simplicity of the finite horizon problem relative to the infinite horizon problem is well-known in the dynamical systems literature (Bertsekas, 1995). In Appendix B we design a learning rule that encourages convergence to the desired state within a small number of time steps τ. Also, the use of symmetric weights is merely sufficient for fixed points to exist, it is not a necessary condition.

We use the terms *Boltzmann-like* and *energy-like* to distinguish our model (with asymmetric weights) from the more strict Boltzmann distribution assumptions. The Boltzmann-like form of the NLCP is,

$$P_B(\mathbf{z}_l|\mathbf{z}_{l-1},\mathbf{z}_{l+1}) = \frac{1}{\zeta(\mathbf{z}_{l-1},\mathbf{z}_{l+1})}\exp\left(-\xi(\mathbf{z}_l,\mathbf{z}_{l-1},\mathbf{z}_{l+1})\right) \quad \text{(NLCP-B)}\ , \tag{2.6}$$

where ξ is the energy-like function and ζ is a normalizing function, with

$$\begin{aligned}\xi(\mathbf{z}_l,\mathbf{z}_{l-1}\mathbf{z}_{l+1}) &= -\mathbf{z}_l^T\mathbf{W}_{l,l-1}\mathbf{z}_{l-1} - \mathbf{z}_l^T\mathbf{L}_l\mathbf{z}_l - \mathbf{z}_l^T\mathbf{W}_{l,l+1}\mathbf{z}_{l+1} - \theta_l^T\mathbf{z}_l \\ \zeta(\mathbf{z}_{l-1},\mathbf{z}_{l+1}) &= \sum_{\mathbf{z}_l}\exp(-\xi(\mathbf{z}_l,\mathbf{z}_{l-1},\mathbf{z}_{l+1}))\ ,\end{aligned} \tag{2.7}$$

where $\mathbf{W}_{l,l+1}$ are top-down weights from layer $l+1$ to l, $\mathbf{W}_{l,l-1}$ are the bottom-up weights from the layer $l-1$ to l, $\mathbf{L}_l$ encodes the influence of units in layer l on other units in that layer (lateral weights), and θ_l is a bias vector. The summation in ζ is over all states of layer l. Note that if the properties of symmetric weights are desired, they can be used without changes to the variational approximation developed below in Section 2.4.

An important question is whether the Boltzmann-like distribution (2.6)-(2.7) is adequate to model the hierarchical sparse generative model of Figure 1. It is possible to construct densities that are not well represented by any set of weights $\mathbf{W},\mathbf{L}$ in (2.7). However, we do not need to model an arbitrary density, only densities that are sparse and therefore have more limited forms of dependence. Algorithms related to the Boltzmann machine have shown success on real-world vision tasks (Teh and Hinton, 2001; Hinton et al., 2006) and tend to confirm that the Boltzmann-like distribution is a reasonable starting point.

2.4 Simplified World Model Developed With a Variational Method

The Boltzmann-like distribution (2.6)-(2.7) provides a reasonable form of the probabilities in the GWM which allows feedforward, feedback and lateral influences. Unfortunately, exact inference on $\mathbf{z}_l$ given $\mathbf{z}_{l-1}, \mathbf{z}_{l+1}$ is intractable for reasonably sized models even when the parameters of $P_B(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ are known because of the need to sum over every possible state $\mathbf{z}_l$ in the normalizing function ζ. In this section, we use a variational method that approximates $P_B(\mathbf{z}_l|\ \mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ with a factorial distribution, $P_Q(\mathbf{z}_l|\ \mathbf{z}_{l-1}, \mathbf{z}_{l+1})$. By *variational* we mean that there are certain parameters $\mathbf{x}_l = \{x_{l,i}\}$ that are varied to make the distribution P_Q as close to P_B as possible. The form of P_Q is taken to be a *generalized factorial Bernoulli distribution*,

$$P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}) = \prod_{i=1}^{s_l} \left[\frac{x_{l,i} - a_4}{a_1}\right]^{\left(\frac{z_{l,i}-a_4}{a_1}\right)} \left[1 - \frac{x_{l,i} - a_4}{a_1}\right]^{\left(1-\frac{z_{l,i}-a_4}{a_1}\right)} \quad , \tag{2.8}$$

where $x_{l,i}$ are the variational parameters and $\mathbf{a} = [a_1, a_2, a_3, a_4]$ are additional constant parameters (a_2 and a_3 will be introduced later) that are used to encourage sparsity-inducing densities (Section 2.5). The dependence on $\mathbf{z}_{l-1}, \mathbf{z}_{l+1}$ will be introduced through $x_{l,i}$ as derived below. A sufficient condition for (2.8) to be a probability distribution is that $\frac{x_{l,i}-a_4}{a_1} + \left(1 - \frac{x_{l,i}-a_4}{a_1}\right) = 1$ and $\frac{x_{l,i}-a_4}{a_1} \geq 0,$ which is true for $a_1 > 0$ and $x_{l,i} \geq a_4$. The slightly generalized Bernoulli-distribution (2.8) is based on a shift in the logical values of $z_{l,i}$ in the energy function from $\{0, 1\}$ to $\{a_4, a_1 + a_4\}$ (the experiments below use $\{-0.05, 1.05\}$, which improves computational efficiency). Our formulation encompasses the two common choices for logical levels, $\{0, 1\}$ and $\{-1, 1\}$, e.g. if logical levels of $\{-1, 1\}$ are needed, then $a_4 = -1, a_1 = 2$. Collecting the $x_{l,i}$ into vectors $\mathbf{x}_l$ of the same size as $\mathbf{z}_l$ for each layer, it can be shown that $\mathbf{x}_l$ are the conditional expected values for each layer,

$$\mathbf{x}_l = E_Q[\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}] \ . \tag{2.9}$$

Note that the variational parameter vector $\mathbf{x}_l$ is the minimum-mean-squared-error (MMSE) estimate of $\mathbf{z}_l$ given the values of its neighboring layers (Kay, 1993, pg. 313).

We now find the $x_{l,i}$ that minimize the Kullback-Leibler divergence (Cover and Thomas, 1991) between the conditional probabilities $P_B(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ and $P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$,

$$\text{KL}(P_Q||P_B) = E_Q[\log P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})] - E_Q[\log P_B(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})] \ \ , \tag{2.10}$$

where E_Q is the expected-value operator with respect to the distribution $P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$. Using the expected value $E_Q[z_{l,i}] = x_{l,i}$, the first term is,

$$\begin{aligned} E_Q[\log P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})] = &\sum_i \left[\frac{x_{l,i} - a_4}{a_1} \log\left(\frac{x_{l,i} - a_4}{a_1}\right)\right. \\ &\left. + \left(\frac{a_1 - x_{l,i} + a_4}{a_1}\right) \log\left(1 - \frac{x_{l,i} - a_4}{a_1}\right)\right] \ . \end{aligned} \tag{2.11}$$

The second term in (2.10) can be expanded,

$$\begin{aligned} E_Q[\log P_B(\mathbf{z}_l|\mathbf{z}_{l-1},\mathbf{z}_{l+1})] &= E_Q[-\log(\zeta) - \xi(\mathbf{z}_l,\mathbf{z}_{l-1},\mathbf{z}_{l+1})] \\ &= E_Q[-\log(\zeta) - \mathbf{z}_l^T\mathbf{W}_{l,l-1}\mathbf{z}_{l-1} - \mathbf{z}_l^T\mathbf{L}_l\mathbf{z}_l - \mathbf{z}_l^T\mathbf{W}_{l,l+1}\mathbf{z}_{l+1} - \theta_l^T\mathbf{z}_l] \ . \end{aligned} \tag{2.12}$$

Again using the expected value $E_Q[z_{l,i}] = x_{l,i}$,

$$\begin{aligned} E_Q[\log P_B(\mathbf{z}_l|\mathbf{z}_{l-1},\mathbf{z}_{l+1})] = &-\log(\zeta) - \sum_{ik} W_{ik}^- \, z_{l-1,k}\, x_{l,i} - \sum_{ik} L_{ik}\, x_{l,k}\, x_{l,i} \\ &- \sum_{ik} W_{ik}^+ \, z_{l+1,k}\, x_{l,i} - \sum_i \theta_{l,i}\, x_{l,i} + c_l \ , \end{aligned} \tag{2.13}$$

where W_{ik}^+, W_{ik}^- and L_{ik} are elements of the weight matrices $\mathbf{W}_{l,l+1}, \mathbf{W}_{l,l-1}$ and $\mathbf{L}_l$ respectively and, defining $\phi_l = (\mathbf{z}_l - \mathbf{x}_l)$, the term $c_l = E_Q[(\mathbf{z}_l - \mathbf{x}_l)^T\mathbf{L}_l(\mathbf{z}_l - \mathbf{x}_l)] = E_Q[\boldsymbol{\phi}_l^T\mathbf{L}_l\boldsymbol{\phi}_l]$, which is zero assuming that $L_{ii} = 0$. [3]

Self-Consistency Conditions of the Variational Approximation. The variational parameters $x_{l,i}$ that minimize the distance between P_B and P_Q (2.10) are found by solving,

$$\begin{aligned} \frac{\partial \mathrm{KL}(P_Q||P_B)}{\partial x_{l,i}} = 0 = a_2 &\left(\sum_k W_{ik}^- z_{l-1,k} + \sum_k L_{ik} x_{l,k} + \sum_k W_{ik}^+ z_{l+1,k} \right) \\ &+ \log\left(\frac{z_1 - x_{l,i} + a_4}{x_{l,i} - a_4} \right) - a_3 \ \ , \end{aligned} \tag{2.14}$$

using a constant term a_3 for the bias $\theta_{l,i}$,[4] and factoring out a_2 from W^+, W^- and L (with a slight abuse of notation, including factoring $\frac{1}{a_1}$ into a_2, a_3, see eq. 2.11). Setting (2.14) equal to zero and solving for $x_{l,i}$,

$$\begin{aligned} x_{l,i} &= f\left(v_{l,i}\right) \\ v_{l,i} &= \sum_k W_{ik}^- z_{l-1,k} + \sum_k L_{ik} x_{l,k} + \sum_k W_{ik}^+ z_{l+1,k} \ \ , \end{aligned} \tag{2.15}$$

where $f(\cdot)$ is a sigmoid activation function parameterized by $\mathbf{a} = [a_1, a_2, a_3, a_4]$,

$$f(v) = \frac{a_1}{1 + \exp\left(-a_2 v + a_3\right)} + a_4 \ . \tag{2.16}$$

Defining $\phi_{l,i} = z_{l,i} - x_{l,i}$ to be the approximation error, the state z is equal to x plus a random noise component, $z_{l,i} = x_{l,i} + \phi_{l,i}$. This yields,

$$\begin{aligned} v_{l,i} &= \sum_k W_{ik}^-(x_{l-1,k} + \phi_{l-1,k}) + \sum_k L_{ik} x_{l,k} + \sum_k W_{ik}^+(x_{l+1,k} + \phi_{l+1,k}) \\ &= \sum_k W_{ik}^- x_{l-1,k} + \sum_k L_{ik} x_{l,k} + \sum_k W_{ik}^+ x_{l+1,k} + \varepsilon_{l,i} \ \ , \end{aligned} \tag{2.17}$$

[3]The term $c_l = E_Q[(\mathbf{z}_l - \mathbf{x}_l)^T\mathbf{L}_l(\mathbf{z}_l - \mathbf{x}_l)] = \mathrm{Tr}[\mathbf{L}_l\Sigma_{\mathbf{z}_l}]$, where $\Sigma_{\mathbf{z}_l}$ is the covariance matrix of $\mathbf{z}_l$ under P_Q. Since $\mathbf{z}_l$ is assumed conditionally independent under P_Q, the non-diagonal elements of the covariance matrix are zero. We will disallow self-feedback (i.e., $L_{ii} = 0$), so that $\mathrm{Tr}[\mathbf{L}_l\Sigma_{\mathbf{z}_l}]$ is zero. However it is straightforward to handle the case when $L_{ii} \neq 0$ given the factorial form of P_Q.

[4]For simplicity we set $\theta_{l,i} = a_3$ for all l, i. However this assumption can be relaxed.

where the ϕ terms have been collected into $\varepsilon_{l,i}$. By collecting all the terms for each layer into a vector we obtain the single equation,

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = f\left(\begin{bmatrix} \mathbf{L}_1 & \mathbf{W}_{12} & 0 & \cdots & 0 \\ \mathbf{W}_{21} & \mathbf{L}_2 & \mathbf{W}_{23} & & \vdots \\ 0 & \mathbf{W}_{32} & \mathbf{L}_3 & \mathbf{W}_{34} & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{W}_{n,n-1} & \mathbf{L}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix} \right) , \tag{2.18}$$

which is the *self-consistency condition* for the variational approximation. The self-consistency condition (2.18) is a necessary condition for the factorial NLCP (2.8) of the SWM to have been optimally fit to the GWM NLCP of (2.6).

Simplified World Model Forms. Further collecting all the estimates for each layer into a single vector $X = [\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T]^T$ and all the weights into a global weight matrix $\mathbb{W}$, equation (2.18) can be written concisely,

$$X = f(\mathbb{W}X + \boldsymbol{\varepsilon}) \qquad \text{(SWM-E)} , \tag{2.19}$$

which is called the *simplified world model on the expected values* (SWM-E), and where the vector forms of the errors are,

$$\begin{aligned} \Phi &= Z - X \\ \boldsymbol{\varepsilon} &= (\mathbb{W} - \mathbb{L})\Phi . \end{aligned} \tag{2.20}$$

The error $\boldsymbol{\varepsilon}$ is generally unobservable, and later we will have to make approximations to perform inference and learn the weights $\mathbb{W}$. In particular, for inference and learning we will neglect $\boldsymbol{\varepsilon}$ and use a *certainty equivalence approximation* (see Section 4). The SWM can be written equivalently in terms of the binary state Z,

$$Z = f(\mathbb{W}Z - \mathbb{L}\Phi) + \Phi \qquad \text{(SWM-B)} , \tag{2.21}$$

and which is called the *SWM on the binary state* (SWM-B). The SWM can also be written in an equivalent dual form on the pre-activation state. Collected the $v_{l,i}$ into a state vector $V \equiv \mathbb{W}X + \boldsymbol{\varepsilon}$ (and $X = f(V)$), we have,

$$V = \mathbb{W}f(V) + \boldsymbol{\varepsilon} \qquad \text{(SWM-P)} , \tag{2.22}$$

which is called the *SWM on the pre-activation state* (SWM-P). Equations (2.19), (2.21) and (2.22) are self-consistency conditions for the SWM. We will return to these key results in Section 3.1 where we discuss how to find solutions to these conditions through evolution of updates in time. Note that with a slight abuse of notation we refer to the self-consistency conditions themselves as the SWMs.

Relation to Other Variational Methods. Our approach is based on using a neighboring-layer conditional probability model which is matched to the hierarchical NLCP GWM using a factorial variational approximation. The use of a factorial variational approximation is known as a *mean-field (MF) approximation* in the statistical physics community (Peterson and Anderson, 1987), where the probabilities to be approximated are either unconditional (as typically done in statistical physics), or else conditioned only on visible layers (as in the Boltzmann machine). A distinguishing feature of our method is that the Boltzmann-like NLCP admits the use an approximating distribution P_Q that is *factorial when conditioned on its neighboring layers*, and the resulting approximation is not factorial when conditioned only on the visible layers. This modeling assumption removes less randomness (and allows more generative capability) than the MF approximation conditioned on the visible layers (as in the deterministic Boltzmann machine, Galland, 1993). This is a richer model than the deterministic Boltzmann machine as our conditional expectations, $\mathbf{x}_l = E_q[\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}]$, retain more randomness than the non-hidden expectations $\mathbf{x}_l = E_q[\mathbf{z}_l|\text{visible layers}]$ of the deterministic Boltzmann machine. Our factorial approximation is reasonable as it is equivalent to saying that the meaningful information about the world contained in any layer is provided by its immediate neighboring layers, so if we condition on neighboring layers then only random ("meaningless") noise remains. The factorial Bernoulli distribution is one of the most tractable and commonly used variational approximations (Jordan et al., 1998), however if more accurate approximations are desired, other distributions may be used such as the second-order methods described by Welling and Teh (2003) and Kappen and Spanjers (2000), although at higher computational cost.

2.5 Activation Functions Can Encourage Sparse Distributions

The parameterized sigmoid activation function (2.16) can be used to encourage sparse activation by appropriate choice of parameters $\mathbf{a}$. Figure 2 shows the activation function (2.16) when parameterized with $\mathbf{a} = [1.1, 2.0, 4.0, -0.05]$, which was chosen so that small levels of activation do not lead to positive values of $f(v)$. Parameters a_2, a_3 can be viewed as prior constraints on the network weights. Theoretically these weight scaling and bias terms could be learned, but from a practical standpoint, our networks are quite large and the critical property of sparsity that makes learning tractable must be enforced from the early epochs, or else too many extra weights would be updated.

We can reasonably assume that $v = v_{l,i}$ as given by (2.15) is a normally distributed random variable due to the central limit theorem (Johnson, 2004) because v is the sum of (nearly) independent, identically distributed values with bounded variance.[5] The density $P(x; \mathbf{a})$ can then be found by transforming the normal density $P(v) = \mathcal{N}(\mu, \sigma^2)$ by the activation function (2.16), see eq. (1.2.26) of Murray (2005). For the values of a given above and for $\mu = 0, \sigma^2 = 1$, Figure 2b shows that $P(x; \mathbf{a})$ is indeed a sharply-peaked

[5]There will be dependence in v between units which all represent the same feature or object, however since all network layers are constrained to be sparse, these dependencies will be much less than the typical pixel-wise dependencies in the original images. Central limit theorems (CLT) that relax the independence assumptions have been developed (Johnson, 2004), and while these extensions are not strictly valid here, they give some level of credence to the belief that CLT-like results should hold in environments with statistical dependencies. The approximate normality of v is confirmed by simulations in Figure 8. Thus, we conclude that assuming normality of v is a reasonable and useful modeling assuming; and one which has been made in other work on large, layered networks (Saul and Jordan, 1998; Barber and Sollich, 2000).

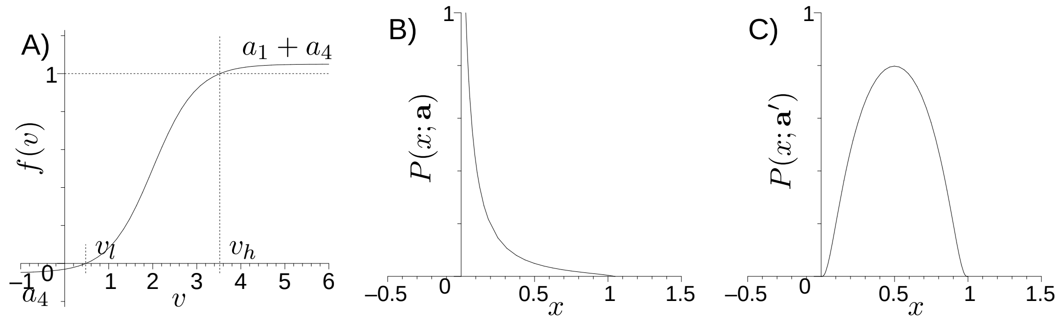


Figure 2: (A) Activation function $f(v)$ with parameters $\mathbf{a} = [1.1, 2.0, 4.0, -0.05]$ (see equation 2.16). The limits of the activation function are $[a_4, a_1 + a_4] = [-.05, 1.05]$, and the slope is set by a_2 and the bias is set by a_3. The shape of the activation function encourages sparsity by ensuring that small input activities $v < v_l$ do not produce any positive output activity. In the simulations, the values of $x = f(v)$ are thresholded so that $x = [f(v)] \in [0, 1]$, however the values of $f'(v)$ are kept for use in the weight updates (see Appendix B). (B) The probability density $P(x; \mathbf{a})$ of a normal random variable ($\mu = 0, \sigma^2 = 1$) after being transformed by the activation function, $f(v)$ in equation 2.16, is a sparsity-inducing density if the parameters $\mathbf{a}$ are chosen properly. The parameters used are $\mathbf{a} = [1.1, 2.0, 4.0, -0.05]$. (C) Probability $P(x; \mathbf{a}')$ is not sparsity-inducing with the standard set of parameters for sigmoid activation functions, $\mathbf{a}' = [1, 1, 0, 0]$.

sparsity-inducing distribution. In contrast, Figure 2c shows $P(x; \mathbf{a}')$ after being transformed by the sigmoid activation function with parameters $\mathbf{a}' = [1, 1, 0, 0]$, which does not lead to a sparsity-inducing distribution. The choice of parameters μ, σ^2 is also important for the transformed distributions to be sparse. For example, if $\mu = 0$, the variance must be less than about 2.0 or else the resulting density will be bimodal. However, with the proper choice of initial conditions, we are able to ensure these conditions are met (see Section 5).

3 Recurrent Dynamic Network

Recognizing that solutions to important inferencing problems correspond to solutions of the self-consistency conditions derived in Section 2.4, we generalize these condition into a dynamic network capable of converging to a solution satisfying (2.18) in order to estimate the states $\mathbf{x}_l$. We introduce a time index t for the iterations of this dynamic network, while our goal remains to estimate the state of the static SWM.

There are n layers in the network and the vector of activations for the l-th layer at time t is denoted $\mathbf{x}_{l,t}, l = 1 \dots n$, with layer sizes $\mathbf{s} = [s_1 \dots s_n]$. The network is designed to enforce rapid convergence to the self-consistency conditions (2.18) for $\mathbf{x}_l$, such that $\mathbf{x}_{l,t} \to \mathbf{x}_l$. The state vector of all the layers at time t is denoted,

$$X_t = \left[\mathbf{x}_1^T \,,\, \mathbf{x}_2^T \,,\, \dots \,,\, \mathbf{x}_n^T\right]^T \in \mathbb{R}^N \quad , \tag{3.1}$$

where N is the size of the state vector (dropping the time index on $\mathbf{x}_l$ inside the vector for clarity). The activity in all layers $\mathbf{x}_l$, is enforced to be sparse and the number of non-zero elements of the layers is denoted $\mathbf{r}_t = [r_1 \dots r_n]$. Figure 3 shows the four-layer network structure used for the experiments in this paper. Dotted lines indicate inputs and connections that are not used here, but are allowed in the model.

Inputs and Outputs The layers used for input and output depend on the type of inference required. In the present work, inputs are usually injected at either the highest or lowest layer (although in general, we may have inputs at any layer if additional types of inference are required). We define an input vector U_t^X (again dropping the time index inside the vector),

$$U_t^X = \left[\mathbf{u}_1^T , \mathbf{u}_2^T , \ldots , \mathbf{u}_n^T\right]^T \quad , \tag{3.2}$$

where $\mathbf{u}_1$ is a sparsely coded input image (see Appendix A) and $\mathbf{u}_n$ is an m-out-of-n binary code called the *object code* which represents the classification of the object. The advantage of using an m-out-of-n object code is that it allows more objects to be represented than the size n of the highest layer, which is the limitation of 1-out-of-n codes. The object code provides a high representational capacity and robustness to the failure of any individual unit/neuron, both of which are desirable from a biological perspective. In addition, we can represent new objects without adjusting the size of the highest layer, $\mathbf{u}_n$, by creating new random object codes.

For recognition and reconstruction, the input $\mathbf{u}_1$ is the coded image, and the object code input is zero, $\mathbf{u}_n = \mathbf{0}$. When the network is used for imagination, the input is the object code presented at the highest layer $\mathbf{u}_n$ and random noise at $\mathbf{u}_2$, and the output is the reconstructed image the lowest layer. For expectation-driven segmentation, both $\mathbf{u}_1$ and $\mathbf{u}_n$ inputs are used. Table 1 shows the layers used for input and output for each type of inference.

3.1 Dynamic Network Form

The recurrent dynamic network (DN-E) is the time-dependent generalization of the self-consistency conditions (2.18) of the SWM-E given by

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}_{t+1} = f\left(\begin{bmatrix} \mathbf{L}_1 & \mathbf{W}_{12} & 0 & \cdots & 0 \\ \mathbf{W}_{21} & \mathbf{L}_2 & \mathbf{W}_{23} & & \vdots \\ 0 & \mathbf{W}_{32} & \mathbf{L}_3 & \mathbf{W}_{34} & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{W}_{n,n-1} & \mathbf{L}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}_t + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}_t \right) + \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \mathbf{u}_n \end{bmatrix}_{t+1} \quad , \tag{3.3}$$

which can be written in the compact form,

$$X_{t+1} = f(\mathbb{W}X_t + \boldsymbol{\varepsilon}_t) + U_{t+1}^X \qquad \text{(DN-E)} \ , \tag{3.4}$$

where U_t^X is the input to the network which can include a sparsely-coded input image $\mathbf{u}_1$ and/or a top-down $\mathbf{u}_n$ consisting of an object code.

Our goal will be to learn a $\mathbb{W}$ such that the network (3.4) will rapidly converge to a steady-state, given transient or constant inputs U_t^X. We will attempt to enforce the steady-state self-consistency behavior at finite time-horizon $t = \tau$, where the horizon τ is a design parameter chosen large enough to ensure that information flows from top-to-bottom and bottom-to-top and small enough for rapid convergence. Because of the block structure of $\mathbb{W}$, information can pass only to adjacent layers during one time step t. (We use the terms *time step* and *iteration* interchangeably.) For example, in a four layer network, it takes only four time steps to propagate information from the highest to the lowest layer, while the network may require more iterations to converge to an accurate estimate. A relatively small number of iterations will be shown to work well, on the order of 8 to 15.

3.2 Pre-Activation State-Space Model

In the previous subsection we created a dynamic network on the state vector X_t based on the SWM-E. By defining an equivalent model on the pre-activation vector V_t, we create another dynamic network which will be used in deriving the learning algorithm (Section 4). Generalizing the pre-activation model (SWM-P, eq. 2.22) to a dynamic network,

$$V_{t+1} = \mathbb{W} f(V_t) + \boldsymbol{\varepsilon}_{t+1} + U^V_{t+1} \qquad \text{(DN-P)} \ , \tag{3.5}$$

where V_t is assumed to be a Gaussian vector as discussed in Section 2.5, and U^V_{t+1} is the input/inital conditions for the pre-activation state (compare with U^X_{t+1} for the state X). The DN-E and DN-P are equivalent representations of a dynamic generative world model. Interpreting the layers of V_t as the hidden states of the generative visual-world model, the visible world is found with the read-out map,

$$Y_t = C\, g(V_t) + \text{noise} \ , \tag{3.6}$$

where $g(\cdot)$ is the output nonlinearity, and $C = [\,\mathbf{1},\, 0,\, \dots\, , 0\,]$ hides the internal states. Table 2 summarizes the moves made from the generative world model of Section 2 to the dynamic networks of the present section.

4 Finding a Cost Function for Learning the Weights $\mathbb{W}$

The dynamic networks of the previous section can perform visual inference by being forced to approximate the self-consistency conditions of the simplified world model (SWM). This can be done assuming that the weights $\mathbb{W}$ are known. Now, we turn to the problem of learning these weights given a set of training data. In this section we will proceed in a Bayesian framework assuming $\mathbb{W}$ is a random variable, [6] and derive a cost function after suitable approximations. The labeled training set is denoted $\check{\mathbb{Y}} = \{\, \check{Y}^{(1)}, \dots, \check{Y}^{(K)}\}$, where the k-th element $\check{Y}^{(k)}$ is a vector with a sparse coding of image k at its first layer $\check{\mathbf{y}}_1^{(k)}$ and the corresponding object code $\check{\mathbf{y}}_n^{(k)}$ at the highest layer and zero vectors at the other layers,

$$\check{Y}^{(k)} = \begin{bmatrix} \check{\mathbf{y}}_1^T & 0 & \dots & 0 & \check{\mathbf{y}}_n^T \end{bmatrix}^T \ , \tag{4.1}$$

where the superscript index of the pattern k for each layer (i.e. $\check{\mathbf{y}}_n^{(k)}$) has been omitted for clarity.

[6] If a non-informative prior on $\mathbb{W}$ is used, this reduces to the maximum likelihood approach.

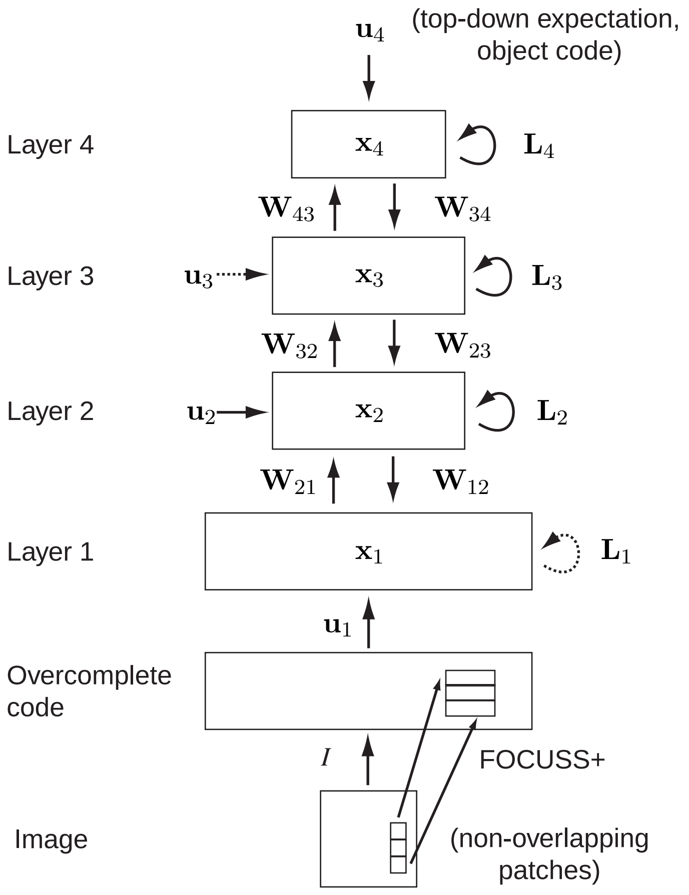


Figure 3: Dynamic network used in the experiments. Inputs images I are first sparsely coded using the FOCUSS+ algorithm, which operates on non-overlapping patches of the input image (Appendix A). This sparse overcomplete code $\mathbf{u}_1$ is used as bottom-up input to the four-layer hierarchical network. Dotted lines indicate inputs ($\mathbf{u}_3$) and connections ($\mathbf{L}_1$) that are not used in the experiments in this paper, but which are allowed by the network.

Table 2: Progression of models developed in Sections 2 and 3.

Hierarchical Generative World Model (GWM)	
Inference given neighboring layers:	
$P(\mathbf{z}_l \mid \mathbf{z}_{l-1}, \mathbf{z}_{l+1})$	(GWM, eq. 2.6)

$$\Downarrow$$

Simplified World Model (SWM) (Self-consistency conditions)	
Variational approximation $E_Q[Z] = X$ leads to:	
$X = f(\mathbb{W}X + \boldsymbol{\varepsilon})$	(SWM-E, eq. 2.19)
Binary state:	
$Z = f(\mathbb{W}Z - \mathbb{L}\Phi) + \Phi$	(SWM-B, eq. 2.21)
Equivalent pre-activation state:	
$V = \mathbb{W}f(V) + \boldsymbol{\varepsilon}$	(SWM-P, eq. 2.22)

$$\Downarrow$$

Dynamic Network (DN) (Discrete-time)	
State update:	
$X_{t+1} = f(\mathbb{W}X_t + \boldsymbol{\varepsilon}_t) + U^X_{t+1}$	(DN-E, eq. 3.4)
Pre-activation state update:	
$V_{t+1} = \mathbb{W}f(V_t) + \boldsymbol{\varepsilon}_{t+1} + U^V_{t+1}$	(DN-P, eq. 3.5)

The cost function for $\mathbb{W}$ is derived using the DN-P dynamics on the pre-activation state V_t (3.5). During training, for each pattern k we create an input time series U^X_t from the data set as follows: $U^X_t = \check{Y}^{(k)}$ for $t = 1, 2, 3$ and $U^X_t = 0$ for $4 \leq t \leq \tau$. This choice of U^X_t starts the dynamic network in the desired basin of attraction for the training pattern $\check{Y}^{(k)}$ ($U^X_t = \check{Y}^{(k)}$ for $t = 1, 2, 3$). The network is then allowed to iterate without input ($U^X_t = 0$ for $4 \leq t \leq \tau$), which with untrained weights $\mathbb{W}$ will in general not converge to the same basin of attraction. The learning process attempts to update the weights so that the training inputs are basins of attraction, and to create middle layer states consistent with that input. The set of inputs for pattern k for all the time steps is denoted $\mathbb{U}^{(k)} = \{U^{(k)}_1 \ldots U^{(k)}_\tau\}$, and for the entire data set we have $\mathbb{U} = \{\mathbb{U}^{(1)} \ldots \mathbb{U}^{(K)}\}$. Similarly, for each pattern in the pre-activation state we have $\mathbb{V}^{(k)} = \{V^{(k)}_1 \ldots V^{(k)}_\tau\}$, and for the whole data set, $\mathbb{V} = \{\mathbb{V}^{(1)} \ldots \mathbb{V}^{(K)}\}$.

Assuming that the weights $\mathbb{W}$ are random variables, their posterior distribution is found by Bayes' rule,

$$P(\mathbb{W}|\mathbb{V};\mathbb{U}) = \frac{P(\mathbb{V}|\mathbb{W};\mathbb{U})P(\mathbb{W})}{P(\mathbb{V};\mathbb{U})} \ . \tag{4.2}$$

Our goal is to find the weights $\mathbb{W}$ that are most likely given the data and the generative model, and we use the *maximum a posteriori* (MAP) estimate,

$$\begin{aligned} \mathbb{W} &= \arg\max_{\mathbb{W}} P(\mathbb{W}|\mathbb{V};\mathbb{U}) \\ &= \arg\min_{\mathbb{W}} -\ln P(\mathbb{V}|\mathbb{W};\mathbb{U}) - \ln P(\mathbb{W}) \ \ , \end{aligned} \tag{4.3}$$

due to the denominator in (4.2) not depending on $\mathbb{W}$. Correct assumptions about $\mathbb{W}$ are important for successful learning, which requires some form of constraint such as prior normalization to use all of the network's capacity (see Section 5). Assuming patterns in the training set are independent, $P(\mathbb{V}|\mathbb{W};\mathbb{U}) = \prod_k P(\mathbb{V}^{(k)}|\mathbb{W};\mathbb{U}^{(k)})$,

$$\mathbb{W} = \arg\min_{\mathbb{W}} \left[-\sum_k \ln P(\mathbb{V}^{(k)}|\mathbb{W};\mathbb{U}^{(k)}) - \ln P(\mathbb{W}) \right] . \tag{4.4}$$

Note that the dynamic system (3.5) is Markovian under our assumption that ε_t are independent (Bertsekas, 1995). Then, the probability of the sequence of time steps can be factored (omitting the pattern index k on the V_t for clarity),

$$\begin{aligned} P(\mathbb{V}^{(k)}|\mathbb{W};\mathbb{U}^{(k)}) &= P(V_\tau, V_{\tau-1}, V_{\tau-2}, \ldots, V_1|\mathbb{W};\mathbb{U}^{(k)}) \\ &= \prod_{t=1}^{\tau} P(V_t|V_{t-1}, \mathbb{W};\mathbb{U}^{(k)}) , \end{aligned} \tag{4.5}$$

from the chain rule of probabilities. The pre-activation state at each time V_t can be expressed in terms of each layer $\mathbf{v}_{l,t}$,

$$P(V_t^{(k)}|V_{t-1}^{(k)}, \mathbb{W};\mathbb{U}^{(k)}) = \prod_{l=1}^{n} P(\mathbf{v}_{l,t}^{(k)}|V_{t-1}^{(k)}, \mathbb{W};\mathbb{U}^{(k)}) , \tag{4.6}$$

if we assume that the layers are conditionally independent of each other at t given the state at the previous time V_{t-1}. Combining (4.4), (4.5) and (4.6),

$$\mathbb{W} = \arg\min_{\mathbb{W}} \left[-\sum_k \sum_{t=1}^{\tau} \sum_{l=1}^{n} \ln P(\mathbf{v}_{l,t}^{(k)}|V_{t-1}^{(k)}, \mathbb{W};\mathbb{U}^{(k)}) - \ln P(\mathbb{W}) \right] . \tag{4.7}$$

Since $\mathbf{v}_{l,t}$ is approximately normal (Section 2.5), for those layers where and when we have target values of $\mathbf{y}_{l,t}$ from the data set and corresponding target states for $\mathbf{v}_{l,t}$,[7] the probability of the layer is,

$$P_{\text{targ}}(\mathbf{v}_{l,t}|V_{t-1}, \mathbb{W};\mathbb{U}) = \frac{1}{(2\pi\sigma_v^2)^{s_l/2}} \exp\left(-\frac{1}{2\sigma_v^2} \boldsymbol{\varepsilon}_{l,t}^T \boldsymbol{\varepsilon}_{l,t} \right) , \tag{4.8}$$

where σ_v^2 is the variance of each component (which is assumed identical). At other layers and times, the state probabilities $\mathbf{v}_{l,t}$ are approximately Gaussian, but we do not have a desired state and so we enforce sparsity in these cases. We model the distributions at these layers by independent Gaussians with fixed mean μ and variance σ_s^2,

$$P_{\text{spar}}(\mathbf{v}_{l,t}|V_{t-1}, \mathbb{W};\mathbb{U}) = \frac{1}{(2\pi\sigma_s^2)^{s_l/2}} \exp\left(-\frac{1}{2\sigma_s^2} \|\mathbf{v}_{l,t} - \boldsymbol{\mu}\|^2 \right) , \tag{4.9}$$

where $\boldsymbol{\mu} = [\mu \ldots \mu]^T$ with μ a design parameter of the appropriate size.

[7]We assume that noise = 0 in (3.6) and that given Y_t we can solve for a corresponding value of V_t.

Introducing an indicator variable β that selects between P_{spar} and P_{targ}, we define $\beta_{l,t} = 1$ if we have target values for layer l at t, $\beta_{l,t} = 0$ otherwise. The probability of each layer becomes,

$$P(\mathbf{v}_{l,t}|V_{t-1}, \mathbb{W}; \mathbb{U}) = \beta_{l,t} P_{\text{targ}}(\cdot) + (1 - \beta_{l,t}) P_{\text{spar}}(\cdot) \ . \tag{4.10}$$

Substituting (4.10) in (4.7) yields,

$$\mathbb{W} = \arg\min_{\mathbb{W}} \left\{ \sum_k \sum_{t=1}^{\tau} \left[\boldsymbol{\varepsilon}_t^T (\boldsymbol{\varepsilon}_t \odot \boldsymbol{\beta}_t) + \lambda (V_t - \boldsymbol{\mu})^T [(V_t - \boldsymbol{\mu}) \odot (1 - \boldsymbol{\beta}_t)] \right] - \ln P(\mathbb{W}) \right\} \ , \tag{4.11}$$

where $\boldsymbol{\beta}_t \in \mathbb{R}^N$ is the indicator vector for all elements of V_t, $\odot$ is the element-wise Hadamard vector product and the constant terms depending on σ_v^2, σ_s^2 have been combined into a new constant λ (again omitting the k inside the summation).

There are several things which should be noted about this formulation. First, the objective function is derived in relation to the pre-activation vector V_t instead of the post-activation vector X_t. This is done to use the Gaussian form of (4.8), and is reminiscent of the technique in the generalized linear model literature of working with the “linear structure vector” of a nonlinear model (Gill, 2001). Secondly, the cost function (4.11) is similar in form to those used in overcomplete coding algorithms, which are unsupervised, and are designed to minimize the reconstruction error using as sparse a code as possible (Olshausen and Field, 1997; Kreutz-Delgado et al., 2003).

The cost function for $\mathbb{W}$ (4.11) is a function of the true state V_t and the error $\boldsymbol{\varepsilon}_t$, which we generally do not have access to. In practice, we will resolve this problem by generating estimates of the unknown V_t using a current estimate of the weights from the dynamic network (DN-P) under the *certainty equivalence approximation* that $\boldsymbol{\varepsilon}_t = 0$ (Bertsekas, 1995). Certainty equivalence is a standard technique in optimization when certain variables are random. For example, an unknown random variable can be replaced by its mean before optimizing the cost function. In our case, we estimate the unknown random V_t by the dynamic network's output $\widehat{V}_t$, which is then used to find the $\mathbb{W}$ that minimizes the cost function (4.11). For each pattern in the data set, we run DN-P (3.5) using the input sequence $U_t^V = v_h U_t^X$, where $v_h = f^{-1}(1.0)$ (Figure 2).[8] Running the network with certainty equivalence gives estimated states,

$$\widehat{V}_t = \mathbb{W} f(\widehat{V}_{t-1}) + U_t^V \ . \tag{4.12}$$

The errors $\widehat{\boldsymbol{\varepsilon}}_t$ used for learning are then the difference between $\widehat{V}_t$ and the desired target states found from the data set,

$$\widehat{\boldsymbol{\varepsilon}}_t = (\widehat{V}_t - v_h \check{Y}^{(k)}) \odot \boldsymbol{\beta}_t \ , \tag{4.13}$$

where layers with no target values are set to 0 due to the effect of $\boldsymbol{\beta}_t$.

Using the cost function in (4.11), we find a learning algorithm for weights $\mathbb{W}$ (Appendix B) which is closely related to the backpropagation-through-time-algorithm (BPTT) for training recurrent networks (Williams and Peng, 1990). The main drawback of the BPTT algorithm is that it is computationally inefficient due to the unrolling of the network for each time step. Our approach overcomes this drawback by using a small number of time steps τ and by taking advantage of the sparsity of every layer to only update weights between units with some non-zero activity.

[8] This is an approximation to $U_t^V = f^{-1}(U_t^X)$ when U_t^X is binary, assuming elements $U_{j,t}^V = 0$ when $U_{j,t}^X = 0$.

5 Algorithm Implementation

This section summarizes the implementation details of the dynamic network and learning algorithm as used in the experiments.

Preparing the Data Set. The data set consists of K images representing M unique objects, where in general we have many different views or transformations of each object, so $K > M$. For each object m, we generate a sparse object code $\mathbf{c}^{(m)} \in \mathbb{R}^{s_n}$ (the size of the highest layer) with r_n randomly-selected non-zero elements, which is used as the desired value of the highest layer. Each image k is preprocessed and converted into a sparse code (Section 6), which is used as the first layer input, $\mathbf{y}_1$. The data set of all images is $\check{\mathbb{Y}} = \{\check{Y}^{(1)}, \ldots, \check{Y}^{(K)}\}$ where each pattern is,

$$\check{Y}^{(k)} = \begin{bmatrix} \check{\mathbf{y}}_1^T & 0 & \ldots & 0 & \check{\mathbf{y}}_n^T \end{bmatrix}^T , \tag{5.1}$$

and the highest layer is the object code, $\check{\mathbf{y}}_n^T = \mathbf{c}^{T(m)}$.

Network initialization. The network weights are initialized with small random values uniformly distributed within certain ranges. The initial weight ranges are: feedforward and feedback weights $W \in [-0.01, 0.01]$, and lateral weights $L \in [-0.001, 0.000]$ (which enables only lateral inhibition, not excitation). Self-feedback is not allowed, $L_{ii} = 0$, and lateral weights are not used in layer 1 for computational efficiency. Feedback weights are initialized to be the transpose of the corresponding feedforward weights, $\mathbf{W}_{lm} = \mathbf{W}_{ml}^T$ but are not restricted to stay symmetric during training.

Performing Inference Given Known Weights $\mathbb{W}$. To run the network for the experiments below, we create an input time series U_t^X from the images and object codes in the data set $\check{\mathbb{Y}}$. The input can include $\check{\mathbf{y}}_1$ and/or $\check{\mathbf{y}}_n$ as determined by the type of inference desired (see Table 1). For example, when the network is run for recognition, the inputs for the first few time steps are the coded image $\check{\mathbf{y}}_1$, so that $(U_t^X)^T = [\check{\mathbf{y}}_1^T, 0, \ldots, 0]^T, t = 1, 2, 3$, and $U_t^X = 0, t \geq 4$. When the network is run generatively, the object code is used as input, such that $(U_t^X)^T = [0, \ldots, \check{\mathbf{y}}_n^T]^T, t = 1, \ldots, \tau$, and the network is then run for τ steps, after which the first layer contains a representation of an imagined image.

Given a sequence of inputs U_t^X the network is run in certainty-equivalence mode (no added noise) for a fixed number of discrete time steps, $0 \leq t \leq \tau$, with τ being 8 to 15 for the experiments below. With an initial state $\widehat{X}_0 = \mathbf{0}$, the network is run using,

$$\begin{aligned} \widehat{V}_t &= \mathbb{W}\widehat{X}_t \\ \widehat{X}_t &= f(\widehat{V}_{t-1}) + U_t^X \qquad 1 \leq t \leq \tau \ . \end{aligned} \tag{5.2}$$

The state $\widehat{X}_t$ is further restricted to be in the unit cube, $\widehat{X}_t \in [0, 1]^N$. To improve computational efficiency, only a limited number of non-zero elements are allowed in each layer, $\bar{\mathbf{r}} = [\bar{r}_1, \ldots, \bar{r}_n]$, which is enforced on V_t at each layer by only allowing the largest $\bar{r}_l$ of them to remain non-zero.

Learning the Weights $\mathbb{W}$. Training proceeds in an online epoch-wise fashion. In each epoch, a subset of patterns is chosen from $\check{\mathbb{Y}}$, and inputs are created with the coded-image in the first layer for the first 3 time steps, so that $U_t^X = [\check{\mathbf{y}}_1^T, 0, 0, \check{\mathbf{y}}_n^T\,], t = 1, 2, 3$, and $U_t^X = 0,\, t \geq 4$. Input patterns must be removed at some point during training because otherwise there would be no error gradient to enforce learning of reconstruction. Presenting the input for 3 time steps was found to give better performance than other lengths of input (see Section 6.1).

The state $\widehat{X}_t$ and pre-activation state $\widehat{V}_t$ from running the network (5.2) are saved for each $t \leq \tau$. The error vector for weight updates is $\widehat{\boldsymbol{\varepsilon}}_t = (\widehat{V}_t - v_h \check{Y}^{(k)}) \odot \boldsymbol{\beta}_t$ (see eq. 4.13). Weight updates Δw_{ji} are given by (B.14). In standard gradient-descent, weight updates will naturally become small when errors are small. However, since we use an additional sparsity enforcing term, even if both the highest and lowest layer errors are zero, weight updates will still occur in order to sparsify middle layers. Training stops after a certain number of epochs have completed.

For computational efficiency when learning sparse patterns, only a small set of weights w_{ji} is updated for each pattern. During our simulations, $\widehat{X}_t$ is found by thresholding the activation function output $f(\widehat{V}_{t-1})$ to $[0, 1]$, resulting in a sparse $\widehat{X}_t$ given certain conditions (Section 2.5). Weights are then only updated between units when the source unit $\widehat{X}_{i,t}$ is active and when either the target unit $\widehat{X}_{j,t}$ is active or has non-zero error $\widehat{\varepsilon}_{j,t}$. [9] During the initial epochs of learning, there must be enough weight strength to cause activation throughout the middle layers. As learning progress, the activity is reduced through the sparseness-enforcing term.

Testing for Classification. To classify an input image once the network has settled into a stable state, the last layer's activation $\mathbf{x}_n$ is compared with the object codes $\mathbf{c}^{(m)}$ to find the class estimate,

$$\text{Class}(\mathbf{x}_n) = \arg \min_{m \in \{1 \ldots M\}} \|\mathbf{x}_n^T - \mathbf{c}^{(m)}\| \quad . \tag{5.3}$$

Weight Normalization. In early experiments with the learning algorithm, we found that some units were much more active than others, with corresponding rows in the weight matrices much larger than average. This suggests that constraints need to be added to weight matrices to ensure that all units have reasonably equal chances of firing. These constraints can also can be thought of as a way of avoiding certain units being starved of connection weights. A similar issue arose in the development of our dictionary learning algorithm (Kreutz-Delgado et al., 2003), and led us to enforce equality among the norms of each column of the weight matrix. Here, both row and column normalization are performed on each weight matrix (feedforward, lateral and feedback). Normalization values are set heuristically for each layer, with an initial value of 1.0 and increasing layer normalization until sufficient activity can be supported by that layer. The normalization values remain constant during network training, and are adjusted from trial to trial.

[9] In theory, the thresholding should not significantly effect the learning. However, due to the size of the network it was not practical to compare thresholded vs non-thresholded performance. Even with the smallest dictionary size (64x64, layer 1 input size = 4096), there are about 5,570,000 weights. Thresholding reduces the actual number of weights updated to about 45,000 per pattern, an increase in speed of over 100x.

6 Visual Recognition and Inference Experiments

In this section, we detail experiments with the learning algorithm developed above and demonstrate four types of visual inference: recognition, reconstruction, imagination and expectation-driven segmentation.

A set of gray-scale images was generated using the Lightwave photorealistic rendering software[10]. Each of 10 objects was rotated 360° through its vertical axis in 2° increments, for a total of $10 \times 180 = 1800$ images, of which 1440 were used for training and the 360 remaining were held out for testing (Figure 4). All images were 64x64 pixels. Before images can be presented to the network they must be sparsely coded which is done with a sequence of preprocessing (Figure 5). First, each image is edge-detected[11] to simulate the on-center/off-center contrast enhancement performed by the retina and LGN. Edge-detected images are then scaled by subtracting 128 and dividing by 256, so that pixel values are $\in [-0.5, 0.5]$. Next, each image is divided into 8x8 pixel patches and sparsely coded with FOCUSS+ using a dictionary learned by FOCUSS-CNDL+ (as described in Appendix A). Dictionaries of size 64x64, 64x128 and 64x196 were learned to compare the effect of varying degrees of overcompleteness on recognition performance. (Figures 6-14 in this section are from experiments with the 64x196 dictionary.) Table 3 shows the accuracy and diversity of the image codes. As dictionary overcompleteness increases from 64x128 to 64x196, both mean-square-error (MSE) and mean diversity decrease, i.e. images are more accurately represented using a smaller number of active elements chosen from the larger overcomplete dictionary. As seen in the third row of Figure 5, the reconstructed images accurately represent the edge information even though they are sparsely coded (on average 192 of 12288 coefficients are non-zero). Finally, the non-negative sparse codes are thresholded to $\{0, 1\}$ binary values before being presented to the network; any value greater than 0.02 is set to 1. This stage, however, does introduce errors in the reconstruction process (last row of Figure 5), and the fidelity of the network's reconstructions will be limited by the binarization. A histogram of coefficient values before binarization is given in Figure 8.

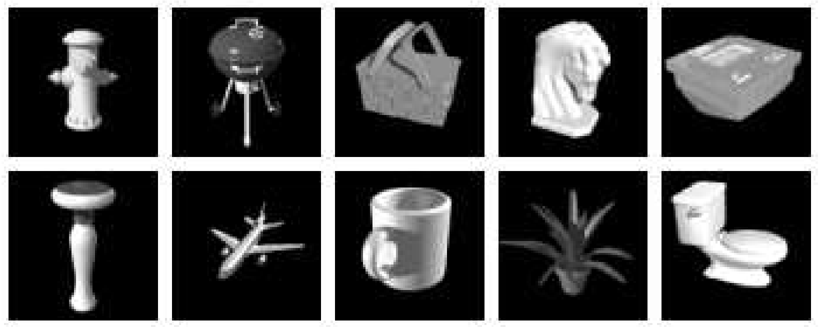
6.1 Recognition with a Four-Layer Network

To test recognition performance, a four-layer network was trained using the data set described above. The training parameters of the network are given in Table 4. Note that all the lateral interactions were forced to be inhibitory or 0, and that no lateral connections were used in the first layer (as we assume the increase in sparsity produced by the FOCUSS+ iterations model the layer 1 lateral connections). Coded images were presented to the first layer of the network for the initial three time steps. Random object codes with $r_4 = 10$ non-zero elements were used on the highest layer. Training took between 11 and 22 hours depending on dictionary size using an Intel Xeon 2.8 Ghz processor. Classification performance reached 100% accuracy on the test set after 135 epochs, but training continued until 1000 epochs to increase the reconstruction accuracy at the first layer. Figure 6 shows the iterations of the network state X_t during classification of a test set image. The first row shows the FOCUSS+ coded input image and the original. The next rows show the activity of each layer and the reconstructed image from the first layer. The object was presented for

[10] Available at `www.newtek.com/products/lightwave/`

[11] Edge detection was done with XnView software (`www.xnview.com`) using the "edge detect light" filter which uses the 3×3 convolution kernel $\begin{bmatrix} 0 & -1 & 0 & ; & -1 & 4 & -1 & ; & 0 & -1 & 0 \end{bmatrix}$.

A)



B)

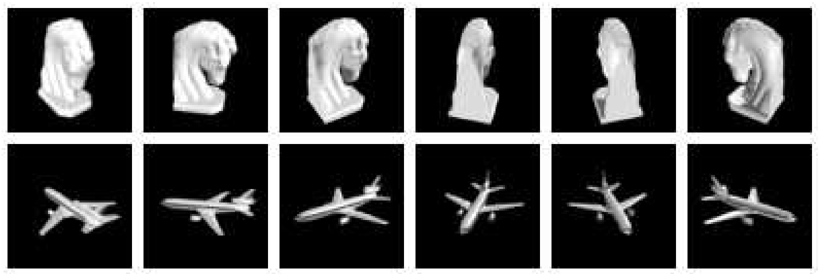


Figure 4: A) Objects used in the experiments, showing one of the 180 views of each object. Images are 64x64 pixel gray-scale. B) Sample rotated object images in the data set.

three time steps and then removed, so that all activity on layer 1 for $t \geq 4$ results from network feedback. As the iterations proceed, the reconstruction completes the outline of the airplane and becomes stronger in intensity. In layer 4, the marker shape indicates whether the unit is active and is part of the correct object code ("■"), or is part of the object code but inactive ("$\circ$"), or is active but should not be ("$\times$"). At $t = 4$, all 10 of the highest layer units in the object code for airplane are active ("■"), so that the image is classified correctly, however there are four other units active that should not be ("$\times$"). At later iterations these extra incorrect units are deactivated (or "sparsified away") so that at $t \geq 5$ only those units in the object code are active, demonstrating the importance of lateral connections in the highest layer. Activity in layers 2 and 3 also decreases with time.

Presenting rotated test-set views of the object shows that the network has learned basins of attraction for the other orientations. Figure 7 shows the state of the network at $t = 7$ after presenting various rotations of the airplane. The invariance of the representation is shown to increase from layer 1 (with nearly completely different units active) through layer 3 (with many of the same units active) to layer 4 (which has identical activity for all four orientations of the airplane).

Training on rotated objects gives the network some robustness to small translations. When tested on images translated +/- 1 pixel in each direction, recognition accuracy is 96.9% on the test set. However, in general, we make no claim that our network has learned transformations (such as translation or scaling) that it has not seen.

The network includes many parameters (Table 4), and learning is more sensitive to some than others. For example, the maximum activity $\bar{r}_2, \bar{r}_3$ for layers 3 and 4 can vary quite widely without noticeable effect

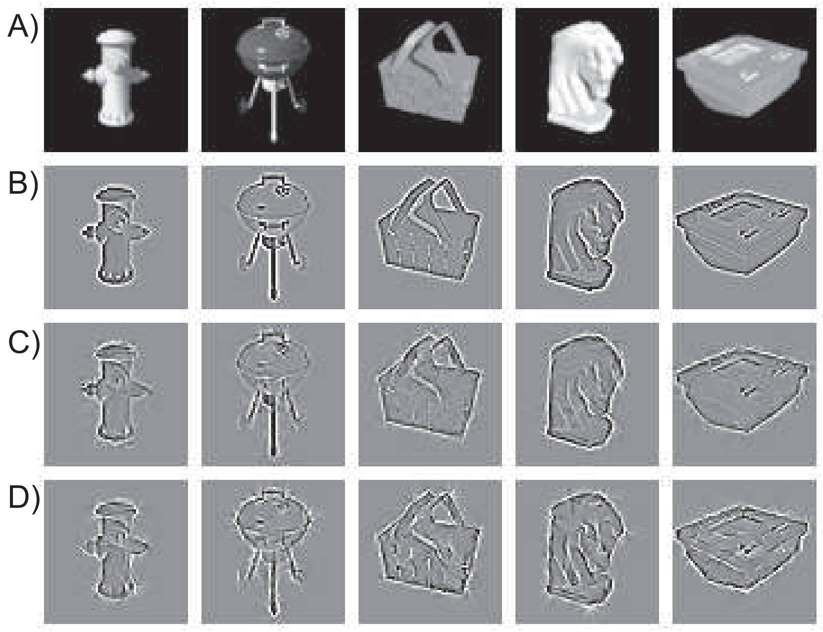


Figure 5: Several preprocessing steps are done before presentation to the network: edge detection, FOCUSS+ sparse coding, and binarization of coefficients. Original images (top row), edge detected images (2nd row), reconstructions from FOCUSS+ codes using a learned overcomplete dictionary (3rd row), and reconstructions from binarized FOCUSS+ codes.

on performance, while increasing $\bar{r}_2$ to 512 (from 100) increases the training time by more than an order of magnitude. This is because weights are only updated between active units, and increasing the maximum number of active units on layer 2 results in a very large number of weights to and from layer 1 that must be updated. When the diversity penalty is turned off ($\lambda = 0$), the average diversity of the 2nd and 3rd layers increases by about 30% and 60% respectively with no significant change in MSE or classification rates. This demonstrates that using the diversity penalty results in more efficient representations (more sparse), consistent with our sparse generative-world model.

We also experimented with different variations of the input time series, and these changes had more dramatic effects on performance. Two experiments were done with 12 time steps: 1) with input presented for 6 time steps and turned off for 6 steps, 2) using a linear decay $1 - t/6$ (input presented at full strength at the first time step and decaying to 0 at the 6th time step). The performance of both of these experiments was worse than the original method (input presented for three steps). For the 6-step input, the recognition accuracy only reached 85% on the test set, for the decaying 6-step input, the accuracy was 90% compared with 100% using the original method. It is not clear why performance drops, but there seems to be a reduction in middle layer activity. Perhaps adjusting normalization or other parameters could improve these results.

To test the Gaussian assumptions made regarding V_t and the errors ε_t, we plot histograms and normal curve fits of randomly chosen units in Figure 8B,C. From Figure 8B we can see that distribution of units

Table 3: Coding performance on 64x64 pixel images (blocked into 8x8 patches) using complete and overcomplete dictionaries. Mean-squared-error (MSE) is calculated over all 8x8 patches in the image, and diversity $= (\#\text{non-zero coefficients})/(\text{layer 1 size})$.

			Diversity		
Dictionary size	Layer 1 size	MSE	Max	Mean	Min
64x64	4096	0.00460	0.0449	0.0266	0.0103
64x128	8192	0.00398	0.0339	0.0240	0.0128
61x196	12288	0.00292	0.0221	0.0156	0.0085

in V_t is quite normal before training. In Figure 8, we see that after training the errors ε_t for layer 1 are less Gaussian, but still reasonably modeled as such. Also, there is more mass in the negative tail indicating patterns where the target values are 1 but the network output is much lower.

6.2 Reconstruction of Occluded Images

Using the same network trained in Section 6.1, reconstruction is demonstrated using occluded images from the test set. Approximately 50% of pixels are set to black by choosing a random contiguous portion of the image to hide. Figure 9 shows the network iterations during reconstruction, where an occluded image is presented for the first three time steps. By $t = 3$, the feedback connections to the first layer have reconstructed much of the outline of the copier object, showing that feedback from the second layer contains much of the orientation-dependent information for this object. Further iterations increase the completion of the outline particularly of the bottom corner and lower-right panel. Another example of reconstruction is shown in Figure 1.13 of Murray (2005).

The network also performs well when recognizing occluded objects. Accuracy is 90% on the occluded test-set objects with the complete dictionary (64x64) and 96-97% with the overcomplete dictionaries. Figure 9 shows that (as above) there are incorrectly activated units in layer 4 at $t = 4$ which are suppressed during later times. In contrast with Figure 6, in layer 2 here there is more activity as time progresses presumably due to the activation of missing features during reconstruction.

More insight into reconstruction can be gained by examining the receptive and projective fields of units in the middle layers. Considering layer 2 (Figure 10), we find the receptive fields (top row) tend to learn a large-scale representation of a particular orientation of an object. This is mainly because the receptive fields are allowed to cover the entire first layer, and no topology is enforced on the weights. Some receptive fields (such as the first column of Figure 10) are tuned to two very different objects, suggesting that units are recruited to represent more than one object, as would be expected from an efficient distributed code. The projective fields are not as clearly specific to a particular orientation and include strong noise, which indicates there must be inhibitory feedback from other layer 2 units contributing to the cleaner version of the layer 1 outputs when the full network is run.

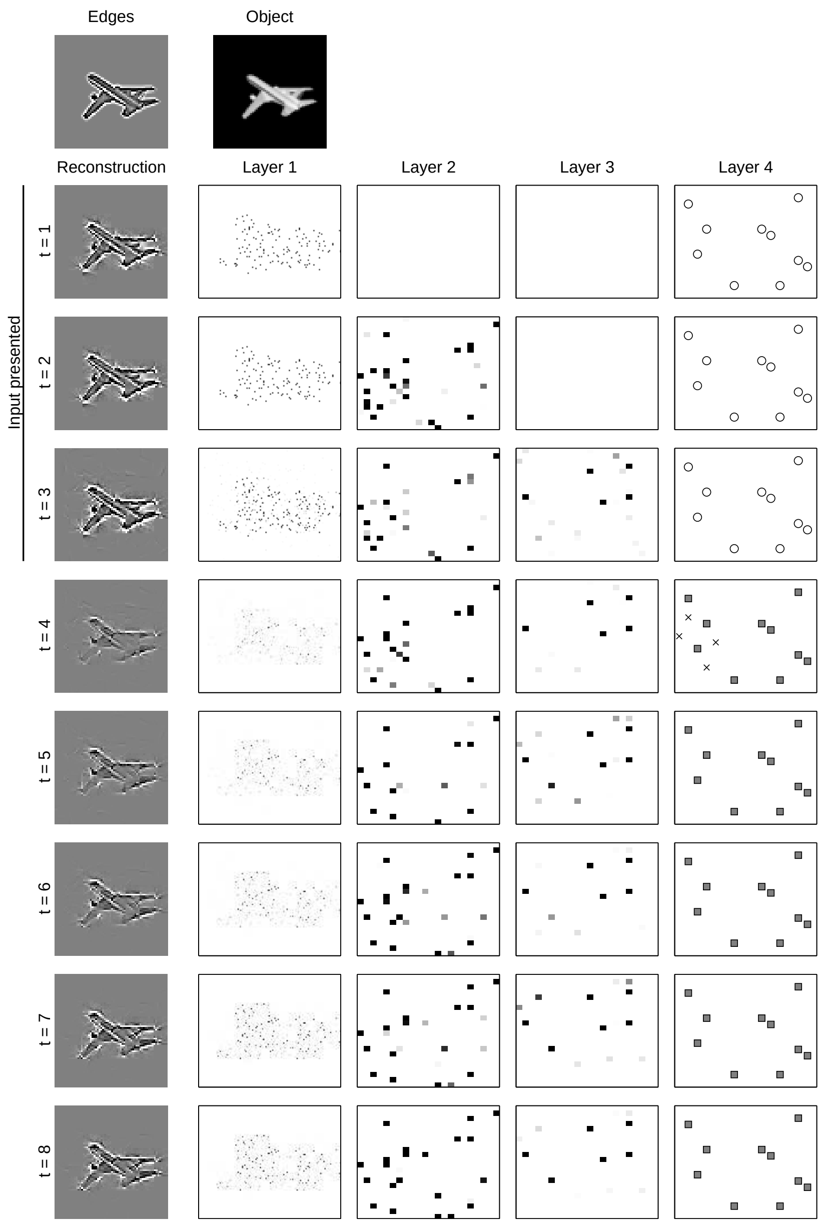


Figure 6: Recognition of a test set object. Each row shows the network activity X_t at a time step. In layer 4, the marker "■" indicates that the unit is active and is part of the correct object code, "$\circ$" that the unit is in the object code but inactive, and "$\times$" that the unit is active but should not be. When $t > 3$, there is no external input and the reconstructed image in layer 1 is due only to network feedback. At $t = 4$ in layer 4 there are four incorrectly activated units ("$\times$") but at later times, the dynamics of the network suppress these incorrectly-active units.

Table 4: Network parameters for training the 4-layer network with 64x196 overcomplete dictionary, corresponding to layer 1 size of 12288. For other sized dictionaries, the size of the first layer was 8192 (64x128 dictionary) and 4096 (64x64 dictionary), with all other parameters as listed below.

Network parameters	
$\mathbf{s}$ (layer size)	[12288, 512, 512, 256]
$\bar{\mathbf{r}}$ (maximum diversity of layer)	[430, 100, 100, 100]
τ (time iterations per pattern)	8
η (learning rate)	0.002
λ (regularization parameter)	0.005
μ (target mean for hidden layers)	-4.0
epoch size (number of patterns)	100
maximum number of epochs	1000
feedforward weight range	[-5.0, 5.0]
feedback weight range	[-5.0, 5.0]
lateral weight range	[-5.0, 0.0]
layer 1 norms (FB)	[12.0]
layer 2 norms (FF, L, FB)	[12.0, 2.1, 2.1]
layer 3 norms (FF, L, FB)	[5.9, 2.1, 1.5]
layer 4 norms (FF, L)	[1.5, 1.5]

6.3 Imagination: Running the Network Generatively

Imagination is the process of running the network generatively with input given as an object code at the highest layer. For this experiment, the network trained in Section 6.1 is used with an object code clamped on the highest layer for all time steps. Random activity is added to the second layer at $t = 3$ so that the network has a means of choosing which view of the object to generate. It was found that increasing the feedback strength (by multiplying feedback weights by 5.0) to the first and second layers increased the activity and quality of the imagined image at the first layer. Without this increase, the layer 1 reconstruction was very likely to settle to the 0 state. Figure 11 shows the results when the object code for the knight is presented. At $t = 4$, the reconstruction is a superposition of many features from many objects but at later times the outline of the object can be seen. The orientation of the generated image alternates between a front view ($t = 5, 7$) and a side view ($t = 6, 8$), which is reminiscent of the bistable percept effect. Not all trials of this experiment result in a bistable state, the majority converged to a single orientation. Interestingly, some orientations of certain objects appear to be generated much more often than other orientations. These "canonical views" represent high probability (low energy) states of the network. A random sample of five imagined objects is shown in Figure 12, showing that a superposition of states can also occur, which is consistent with the projective field properties shown in Figure 10.

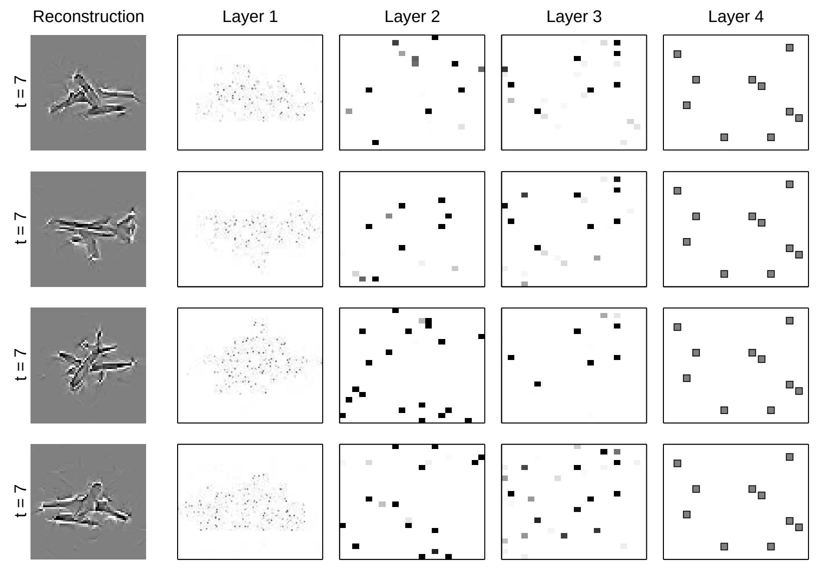


Figure 7: Each row is the network state X_t at $t = 7$ after presenting various rotated images of the airplane (test set images, views unseen during training), demonstrating that multiple basins of attraction can be learned for each object. Higher layers show more invariant representations than lower layers, with layer 4 showing the fully-invariant representation of the airplane.

6.4 Expectation-Driven Segmentation: Out from Clutter

In expectation-driven inference, both an input image and a top-down expectation are presented to the network, and the output can either be the highest-layer classification or the lowest-layer reconstructed image. Here, we considered the later case where the desired output is a segmented image reconstructed from the first layer. The same network trained in Section 6.1 is used here with increased feedback strength as described in Section 6.3. Cluttered input images are created by combining many objects from the data set at random translations, overlayed with a portion of the desired image (the same portion, 50%, used in the reconstruction experiment). This is a fairly difficult recognition problem as the clutter in each image is composed of features from trained objects, so that competing features tend to confound recognition algorithms. Although the features from the clutter objects are likely to be in different locations than seen during training, it is still a more difficult task than segmentation from a randomly chosen (untrained) background.

The problem of expectation-driven segmentation is different from recognition in that we ask the network not “what object is this?” but “assuming object x is here, what features in the image most likely correspond to it?” For this experiment, we present at $t = 2, 3$ the image of the occluded object in clutter and at $t = 1, \ldots, 4$ the expectation that the object is present at the highest layer. Figure 13 shows the network states when presented with a cluttered image and top-level expectation of the knight object. The timing of the inputs was arranged so that the feedback and feedforward input first interact at $t = 3$ in layer 3. When $t = 4$, the input image is no longer presented and the network feedback has isolated some features of the object. Later time steps show a sharper and more accurate outline of the knight, including edges that were occluded in the input image. At the highest layer, feedforward interactions from lower layers cause the correct object code to degrade. At $t = 12$ all the units in the object code for knight were active, as well as

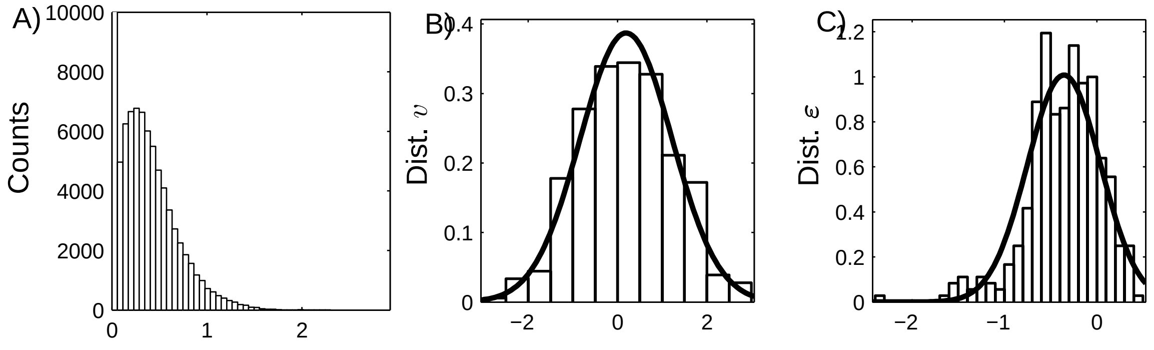


Figure 8: A) Histogram of FOCUSS+ coefficient values before binarization. There 2.9×10^6 elements in the zero bin, and the maximum coefficient value is 2.9. B) Histogram and Gaussian fit of a randomly chosen unit in the preactivation state V_t in layer 3 before training. C) Histogram and Gaussian fit of residual ε_t in layer 1 after training.

four incorrectly active units, which still allows correct classification. To illustrate the need for the top-down expectation input in this case, Figure 14 shows the states at $t = 1, 4, 8$ when no object code is presented at layer 4. The activity gradually decays and there is no reconstruction at layer 1. Comparing Figure 11 (imagination) and Figure 13 shows that the partial information provided in the cluttered image is enough to keep the network at a stable estimate of segmentation, and in this case prevent oscillations between two orientations (which occurred when only top-down input was present).

6.5 Overcompleteness Improves Recognition Performance

One of the central questions addressed in this work is how a sparse overcomplete representation in the early stages of visual processing, e.g. V1 in primates (Sereno et al., 1995), could be useful for visual inference. As described above, we trained networks using learned dictionaries of varying degrees of overcompleteness: 64x64, 64x128 and 64x196, and corresponding sizes of the first layer: 4096, 8192 and 12288. Performance was compared on the test set objects, occluded objects, and objects in clutter. The cluttered images were created by overlaying the entire object on a cluttered background, resulting in a somewhat easier problem than the occluded-object-in-clutter images used in Section 6, although here no top-down expectations were used to inform the recognition. Figure 15 shows the recognition accuracy on these three image sets. For the test set (complete images), all three networks had performance at 99-100%, but for the occluded and cluttered images there is a gain in accuracy when using overcomplete representations, and the effect is more pronounced for the more difficult cluttered images. For occluded objects, accuracy was 90% (324/360) for the complete dictionary and 97% (349/360) for the 3x overcomplete dictionary. The most significant improvement was with the cluttered images; accuracy was 44% (160/ 360) for the complete dictionary, and 73% (263/360) for the 3x overcomplete dictionary. While the absolute classification rate for the cluttered images might appear low (44-73%), many of the misclassified objects were those of smaller size (e.g. the airplane and fire-hydrant) which allowed more features from other larger objects to be visible and confound the recognition. In addition, neither the dictionary nor the network were trained on images with clutter, so

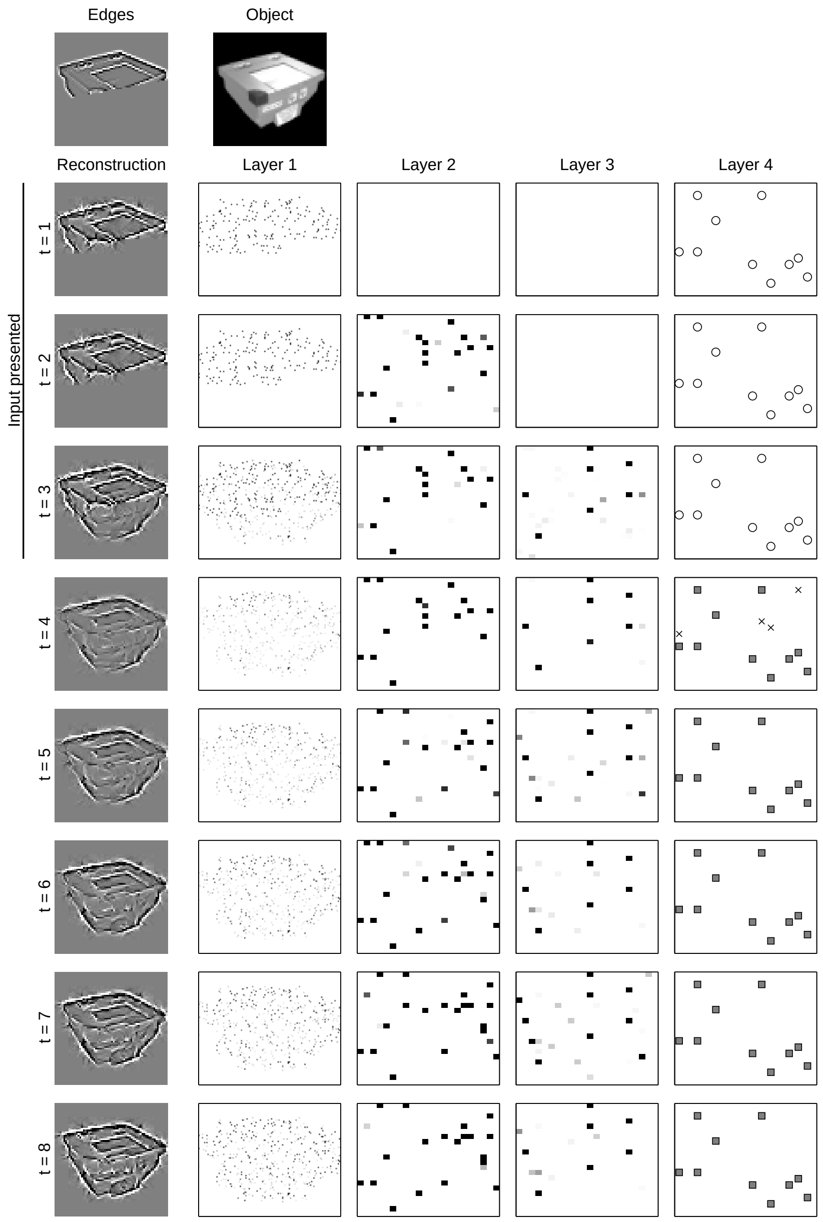


Figure 9: Reconstruction of an occluded input image. As early as $t = 3$, feedback from layer 2 results in reconstruction of some of the outer edges of the objects. More detail is filled in at later time steps. Layer 4 legend: "■" = unit is active and in correct object code, "○" = unit is in the object code but inactive, "$\times$" = unit is active but should not be (not in object code).

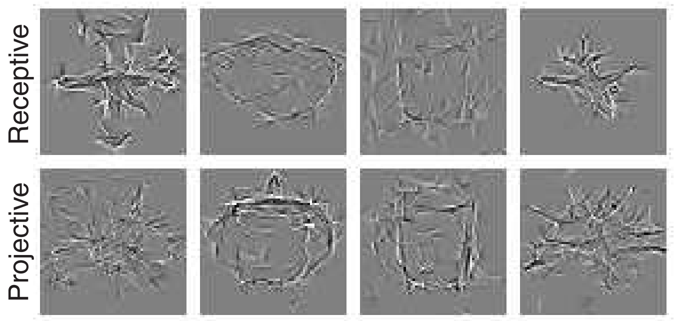


Figure 10: Receptive and projective fields of four units in layer 2. For each unit, the top row shows the receptive fields (feedforward weights from layer 1 to 2), and bottom row shows the projective field (feedback weights from layer 2 to 1). The weight vectors are converted into images by multiplying by the learned dictionary. The unit in the first column is tuned to respond to both the plane and the table, while its projective field appears to include many possible orientations of the plane.

the network had no previous experience with this particular type of cluttered images.

7 Discussion

In this section we discuss the motivations for our network and compare it with other recurrent and probabilistic models of vision. Additional discussion can be found in Murray (2005, Sec. 1.8).

7.1 Why Sparse Overcomplete Coding and Recurrence?

In the brain, early visual areas are highly overcomplete, with about 200-300 million neurons in V1 compared to only about 1 million neurons that represent the retina in the lateral geniculate nucleus (LGN) of the thalamus (Stevens, 2001; Ejima et al., 2003). As primate evolution has progressed, there has been an increase in the ratio of V1 to lateral geniculate nucleus (LGN) size. While even the smallest of primates shows a high degree of overcompleteness, the increase in higher primates is linked with increase in retinal resolution and presumably improved visual acuity (e.g. 87x overcomplete for the tarsier monkey compared with 200-300x for humans).

Mathematically, sparse coding strategies are necessary to make efficient use of overcomplete dictionaries because the dictionary elements are generically non-orthogonal. To provide a low-redundancy representation (Attneave, 1954; Barlow, 1959) a sparse set of elements must be chosen that accurately represents the input. If we have faith in the generative model postulated in Figure 1, real-world images can be accurately modeled as being caused by a small number of features and objects, supporting the choice of a sparse prior (even in the case of complete coding). Other benefits of sparse coding include: making it easier to find correspondences and higher order correlations, increasing the signal-to-noise ratio, and increasing the storage and representational capacity of associative memories (Field, 1994). Biological evidence for sparse coding ranges from the simple fact that average neural firing rates are low, 4-10 Hz (Kreiman et al., 2000), to experiments that find sparseness in V1 increases as larger patches of natural images are presented indicating that a

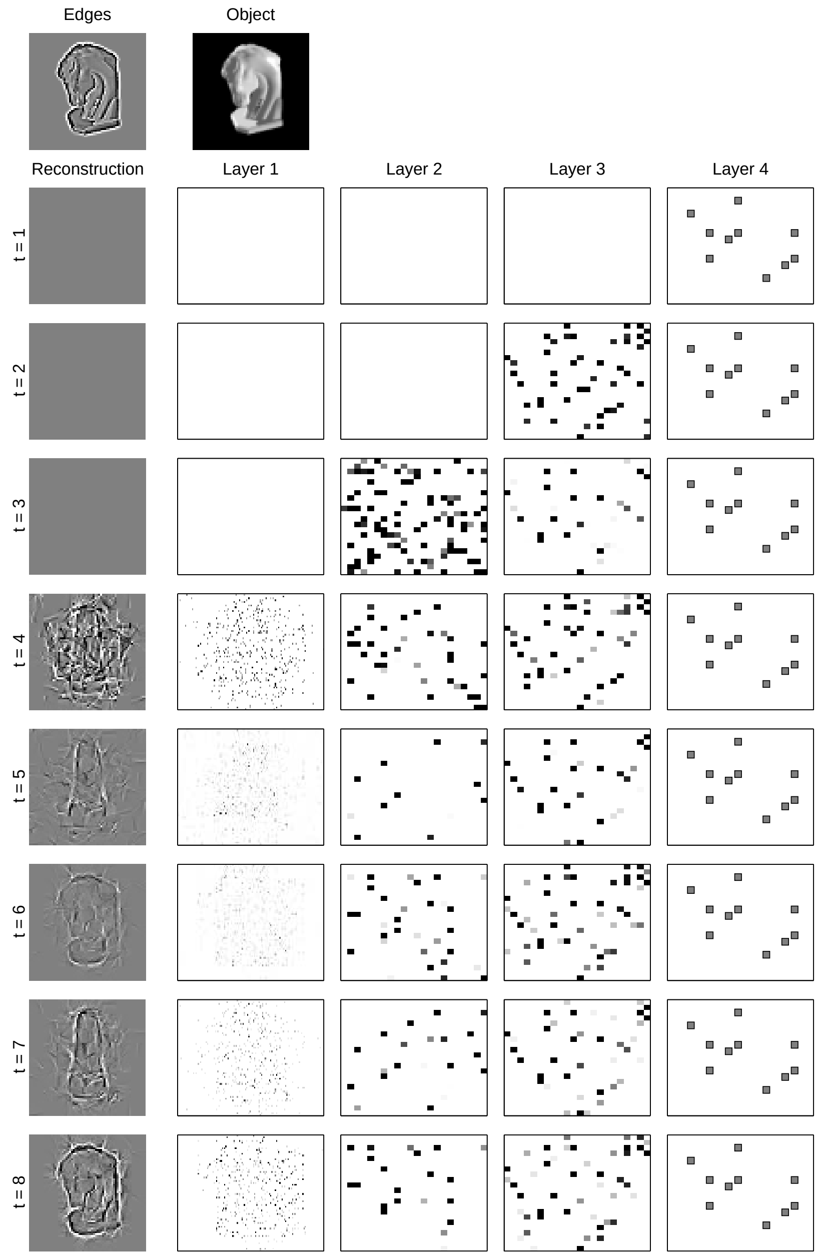


Figure 11: Imagination using the object code for the knight as the top-down input and the injection of random activity in layer 2 at $t = 3$. The reconstruction is a bistable (oscillating) pattern of the object from the front and side views.

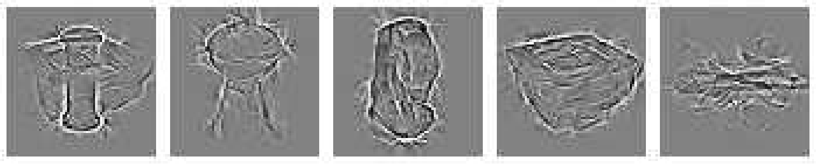


Figure 12: Imagination using random object codes as input to layer 4 and at $t = 3$, random activity at layer 2. The images are the network's layer 1 state at $t = 15$ with top-layer objects codes of the fire-hydrant, grill, knight, copier and airplane. For the first and last images, the network has settled into a superposition of multiple objects (fire-hydrant and copier) or multiple orientations of the same object (airplane).

concise representation can be found by deactivating redundant features, presumably through the interaction of lateral and feedback inhibition (Vinje and Gallant, 2000). One of the successes of sparse-coding theory has been the learning of receptive fields that resemble the orientation and location selectivity of V1 neurons (Olshausen and Field, 1997), and extensions have been made to model complex cells (Hoyer and Hyvärinen, 2002).

While overcompleteness and sparse-coding are important features of early vision in V1, perhaps the most striking aspect of higher visual areas is the amount of lateral and feedback connections within and between areas (Felleman and Van Essen, 1991). Even in V1, lateral and feedback input from other cortical areas account for about 65% of activity, with only 35% of response directly due to feedforward connections from the LGN (Olshausen and Field, 2005). We showed in Section 2 that feedback and lateral connections are required for many types of inference. In some recognition tasks, there is evidence that the brain is fast enough to complete recognition without extensive recurrent interaction (Thorpe et al., 1996). Consistent with this, our model is capable of quickly recognizing objects in tasks such as Figure 6, where the correct object code is found at $t = 5$. However, more difficult tasks such as segmentation (Figure 13) require recurrence and would take longer for the brain (Lee et al., 1998).

7.2 Related Work: Biologically Motivated Models of Vision

There have been many hierarchical models created to explain vision, and these fall into two main categories: feedforward-only or recurrent (which include various types of feedback and lateral connections between layers). Some examples of the feedforward class are the Neocognitron model of Fukushima and Miyake (1982); VisNet of Rolls and Milward (2000); and the invariant-recognition networks of Földiák (1991) and Riesenhuber and Poggio (1999). While many of these models use sparsity with some form of winner-take-all competition which is usually interpreted as lateral interaction, since they do not include feedback connections they are not capable of the range of inference described in Section 2.2, and will not be discussed further here.

One of the more closely related works is the dynamic network developed by Rao and Ballard (1997). A stochastic generative model for images is presented and a hierarchical network is developed to estimate the underlying state. Their network includes multiple layers with feedforward and feedback connections which are interpreted as passing the residuals from predictions at higher levels back to lower levels (but with no explicit learnable lateral connections, nor overcomplete representations). Experiments demonstrate

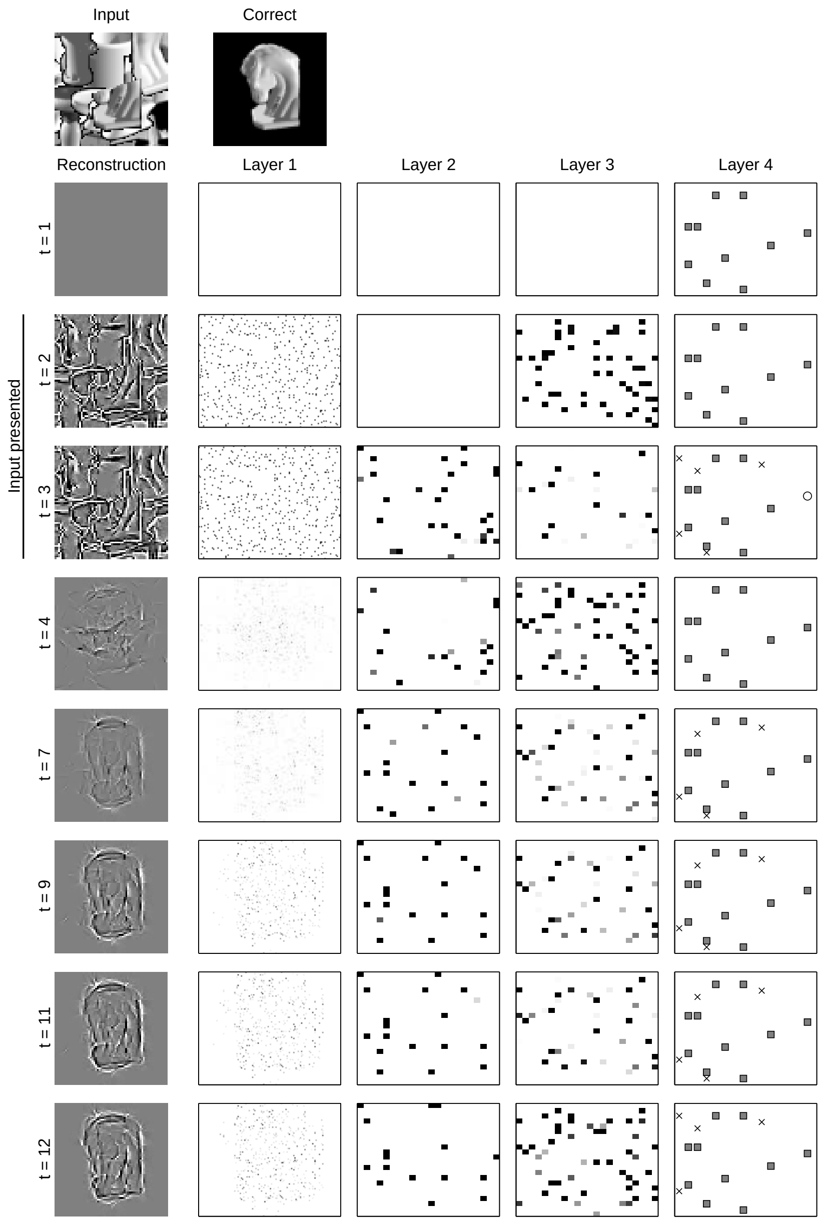


Figure 13: Expectation-driven segmentation using occluded objects over a cluttered background. The clutter input is presented at the lowest layer for $t = 2, 3$. Top-down expectations (the object code for knight) are presented at the highest layer for $t = 1, \ldots, 4$. By $t = 12$, the network converges to a segmented outline of the knight in the correct orientation at the first layer. Layer 4 legend: "■" = unit is active and in correct object code, "○" = unit is in the object code but inactive, "×" = unit is active but should not be (not in object code).

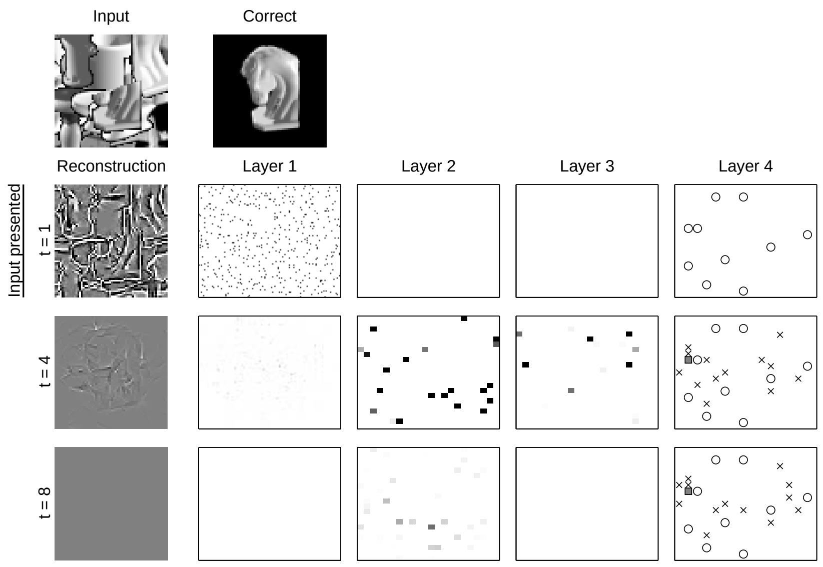


Figure 14: Recognizing the occluded object in a cluttered background is difficult without top-down expectations. The same input image used in Figure 13 is presented for $t = 1, 2, 3$, however no top-down inputs are present. A few representative time steps show that the activity gradually decays over time, and no object is reconstructed at layer 1. Layer 4 legend: "■" = unit is active and in correct object code, "$\circ$" = unit is in the object code but inactive, "$\times$" = unit is active but should not be (not in object code).

recognition, reconstruction of occluded images, learning of biologically plausible receptive fields and ability to tell that an object had not been seen during training. Perhaps because of the computational requirements, only fairly limited recognition experiments were performed, using five objects (one orientation per object) and rotation invariant recognition with two objects, each with 36 views used for training and testing (Rao, 1999).

Newer versions of the Neocognitron include feedback connections and are demonstrated for recognition and reconstruction (Fukushima, 2005). The model posits two types of cells in each region of the system, S-cells and C-cells in analogy with the simple and complex cells categorized by Hubel and Wiesel (1959). The S-cells are feature detectors and the C-cells pool the output of S-cells to create invariant feature detectors. To solve the reconstruction problem, further cell types and layers are added, and many of the layers have different learning rules. In contrast, our network is able to perform various inference types without changes to the architecture or learning rule.

7.3 Related Work: Probabilistic Models in Computer Vision

Recent work in computer vision has investigated probabilistic generative models apart from direct biological motivation (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Fergus et al., 2007; Sudderth et al., 2005).

Most closely related to our work is the learning algorithm of Hinton et al. (2006) for hierarchical belief networks. The network has multiple hidden layers followed by a much larger overcomplete associative memory (whereas our overcomplete stage occurs at the second layer), and a highest layer with a 1-out-of-n

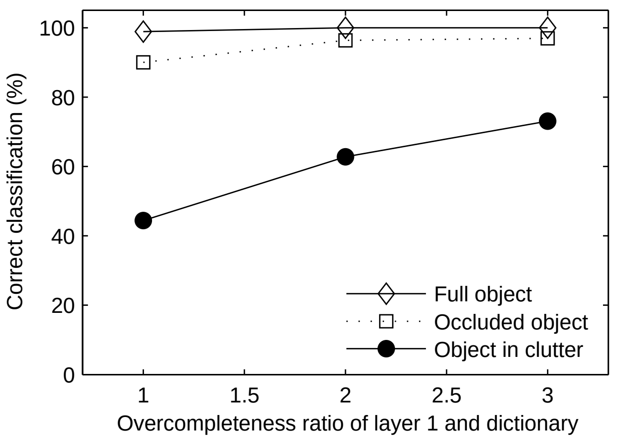


Figure 15: Recognition performance on the test set (full object), occluded images (50% occlusion) and cluttered images with three different degrees of overcompleteness in Layer 1 representation and learned dictionaries. Recognition performance improves with increased overcompleteness, particularly in the difficult cluttered scenes. Test set size is 360 images (36 views of 10 objects).

code for the object class. The first layer has real-valued inputs, while stochastic binary values are used at higher layers. Feedforward and feedback weights are learned but no lateral connections are used, and during testing only one forward-backward pass is made at each layer. When trained on a benchmark handwritten digit data set, the accuracy is competitive with the best machine learning methods, showing that generative hierarchical networks are promising for real-world vision tasks. Using a similar learning procedure with an autoencoder network architecture, Hinton and Salakhutdinov (2006) show applications to data compression and text classification.

While there are many differences between this work and our algorithm, they address the same basic question of how to train hierarchical generative models. One important difference is that Hinton et al. (2006) use stochastic units and Gibbs sampling for generative inference, while we use a nearest-layer conditional variational approximation. We believe the factorial approximation of eq. (2.8) can be sufficiently accurate in the case of sparse activations, and that enforcing a short enough time-horizon τ makes learning computationally tractable. More experiments with known generative models will be needed to further evaluate the differences between these algorithms.

Fergus et al. (2007) develop a model for classification of object categories in unsegmented images. The first step is finding a small set of interesting features using a saliency detector. For each category, a probabilistic model is learned for these features including their relative position and scale. Impressive detection performance is achieved on real-world data sets. In contrast to our work which models all the features in the image, Fergus et al. (2007) use only a small number of features (< 30), so that, if run generatively, their model would only reconstruct a small subset of the features in each object. Using a saliency detector improves position and scale invariance (which would benefit our network), however, using only this small feature set reduces performance when features of a class model cannot be found.

In a related work, Sudderth et al. (2005) present a probabilistic model of object features and apply it to object categorization in real-world scenes. Similar to our model and in contrast with Fergus et al. (2007), their model is a true multiclass classifier which allows features to be shared between models of different objects, and allows for more rapid classification without the need to run multiple classifiers. As above, the small feature set limits the potential detail of generative reconstruction. However, segmentation results show that regions such as “building”, “car” and “street” can be detected in city scenes.

8 Conclusions

We have developed a framework and learning algorithm for visual recognition and other types of inference such as imagination, reconstruction of occluded objects and expectation-driven segmentation. Guided by properties of biological vision, particularly sparse overcomplete representations, we posit a stochastic generative world model. Visual tasks are formulated as inference problems on this model in which inputs can be presented at the highest layer, lowest layer, or both depending on the task. A variational approximation (the simplified world model) is developed for inference which is generalized into a discrete-time dynamic network.

An algorithm is derived for learning the weights in the dynamic network, with sparsity-enforcing priors and error-driven learning based on the pre-activated state vector. Experiments with rotated objects show that the network dynamics quickly settle into easily-interpretable states. We demonstrate the importance of top-down connections for expectation-driven segmentation of cluttered and occluded images. Four types of inference were demonstrated using the same network architecture, learning algorithm and training data. We show that an increase in overcompleteness directly leads to improved recognition and segmentation in occluded and cluttered scenes. Our intuition as to why these benefits arise is that overcomplete codes allow the formation of more basins of attraction and higher representational capacity.

Acknowledgments

J.F.M. gratefully acknowledges support from the ARCS Foundation. This research was supported in part by NSF cooperative agreement ACI-9619020 through computing resources provided by the National Partnership for Advanced Computational Infrastructure at the San Diego Supercomputer Center. Thanks also to Virginia de Sa, Robert Hecht-Nielsen, Jason Palmer, Terry Sejnowski, Sebastian Seung, Tom Sullivan, Mohan Trivedi, and David Wipf for comments and discussions.

A Sparse Image Coding with Learned Overcomplete Dictionaries

The dynamic network and learning algorithm presented above require that the inputs $\mathbf{u}_l$ at each layer be sparse vectors. To transform the input image into a suitable sparse vector, we use the *focal underdetermined-system-solver* (FOCUSS) algorithm for finding solutions to inverse problems. The FOCUSS algorithm represents data in terms of a linear combination of a small number of vectors from a dictionary, which may be overcomplete. Other methods for sparsely-coding signals include matching pursuit, basis pursuit, and sparse Bayesian learning, which were also evaluated for image coding (Murray and Kreutz-Delgado, 2006). The overcomplete dictionary is learned using the FOCUSS-CNDL (column-normalized dictionary learning) algorithm developed by Murray and Kreutz-Delgado (2001); Kreutz-Delgado et al. (2003).

The problem that FOCUSS-CNDL addresses here is that of representing a small patch of an image $\mathbf{y} \in \mathbb{R}^m$ using a small number of non-zero components in the source vector $\mathbf{x} \in \mathbb{R}^n$ under the linear generative model,

$$\mathbf{y} = A\mathbf{x}\,, \tag{A.1}$$

where the dictionary A may be overcomplete, $n \geq m$. The algorithm updates and more discussion of the FOCUSS-CNDL algorithm in this context are given in Murray (2005, Section 1.A). Parameters for FOCUSS-CNDL are: data set size = 20000 image patches, block size $N = 200$, dictionary size = 64x64, 64x128, or 64x196, diversity measure $p = 1.0$, regularization parameter $\lambda_{max} = 2 \times 10^{-4}$, learning rate $\gamma = 0.01$, number of training epochs = 150, reinitialization every 50 epochs. After each dictionary update, A is normalized to have unit Frobenius norm, $\|A\|_F = 1$ and equal column-norms. Figure 1.18 of Murray (2005) shows the learned 64x196 dictionary after training on edge-detected patches of man-made objects (the data set described in Section 6).

Once the dictionary A has been learned, input images for the dynamic network (DN) are coded using the FOCUSS+ algorithm (Murray and Kreutz-Delgado, 2006). The input images are divided into consecutive non-overlapping patches of the same 8x8 size used for dictionary learning. The FOCUSS+ algorithm consists of repeated iterations of eq. 6 from Murray and Kreutz-Delgado (2006) over an image patch y_k to estimate x_k. Each x_k is updated for 15 iterations with $p = 0.5$.

B Derivation of Learning Algorithm for $\mathbb{W}$

The learning algorithm for the weights $\mathbb{W}$ is derived similarly to the backpropagation-through-time-algorithm (BPTT) (Williams and Peng, 1990). Using the pre-activation cost function (4.11) for an individual pattern,

$$J_{PA} = \frac{1}{2}\sum_{t=1}^{\tau}\left[\boldsymbol{\varepsilon}_t^T(\boldsymbol{\varepsilon}_t \odot \boldsymbol{\beta}_t) + \lambda(V_t - \boldsymbol{\mu})^T[(V_t - \boldsymbol{\mu}) \odot (1 - \boldsymbol{\beta}_t)]\right]\ , \tag{B.1}$$

which uses states V_t generated from running the network in certainty-equivalence mode. The effect of the weight prior $-\ln P(\mathbb{W})$ will not be considered in this section, as it was found that enforcing periodic weight normalization is more computationally efficient than using prior constraints in every weight update (see Section 5).

To minimize the cost J_{PA} we update the weights using gradient descent,

$$\Delta w_{ji} = -\eta\frac{\partial J_{PA}}{\partial w_{ji}} = -\eta\sum_{t=1}^{\tau}\frac{\partial J_{PA}}{\partial V_{j,t}} \cdot \frac{\partial V_{j,t}}{\partial w_{ji}}\ , \tag{B.2}$$

where w_{ji} is the element from the j-th row and i-th column of $\mathbb{W}$. The second term on the right is,

$$\frac{\partial V_{j,t}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\mathbb{W}_{j\cdot}\,X_t = X_{i,t}\ , \tag{B.3}$$

where $\mathbb{W}_{j\cdot}$ is the j-th row of $\mathbb{W}$. The first term on the right of (B.2) is divided into two parts,

$$\begin{aligned}\frac{\partial J_{PA}}{\partial V_{j,t}} &= B_{j,t} + D_{j,t} \\ B_{j,t} &= \frac{\partial}{\partial V_{j,t}}\left[\frac{1}{2}\sum_{\rho=1}^{\tau}\boldsymbol{\varepsilon}_\rho^T(\boldsymbol{\varepsilon}_\rho \odot \boldsymbol{\beta}_\rho)\right] \\ D_{j,t} &= \frac{\partial}{\partial V_{j,t}}\left[\frac{\lambda}{2}\sum_{\rho=1}^{\tau}(V_t - \boldsymbol{\mu})^T[(V_t - \boldsymbol{\mu}) \odot (1 - \boldsymbol{\beta}_t)]\right] \quad ,\end{aligned} \tag{B.4}$$

where B is related to reconstruction error and D increases sparsity on those layers without desired values. Recursion expressions can now be found for $B_{j,t}$ and $D_{t,j}$. First, some notation: the j-th row of the weight matrix $\mathbb{W}$ is denoted $\mathbb{W}_{j\cdot}$ and the element from the j-th row and i-th column is w_{ji}. Beginning with the reconstruction-enforcing term B (temporarily omitting the binary indicator variable $\boldsymbol{\beta}$ for notational clarity),

$$B_{j,t} = \frac{\partial}{\partial V_{j,t}}\left[\frac{1}{2}\sum_{\rho=1}^{\tau}\boldsymbol{\varepsilon}_\rho^T\boldsymbol{\varepsilon}_\rho\right] \quad . \tag{B.5}$$

For $B_{j,t}$, at the last time step in (B.2) when $t = \tau$ only the $\rho = \tau$ terms depend on $V_{j,\tau}$,

$$B_{j,\tau} = \frac{\partial}{\partial V_{j,\tau}}\left[\frac{1}{2}\boldsymbol{\varepsilon}_\tau^T\boldsymbol{\varepsilon}_\tau\right] = \boldsymbol{\varepsilon}_\tau^T\frac{\partial}{\partial V_{j,\tau}}\boldsymbol{\varepsilon}_\tau = \varepsilon_{j,\tau} \quad , \tag{B.6}$$

where $\varepsilon_{j,t}$ is the j-th element of the error vector $\boldsymbol{\varepsilon}_t$. When $t = \tau - 1$,

$$B_{j,\tau-1} = \frac{\partial}{\partial V_{j,\tau-1}}\left[\frac{1}{2}\boldsymbol{\varepsilon}_\tau^T\boldsymbol{\varepsilon}_\tau + \frac{1}{2}\boldsymbol{\varepsilon}_{\tau-1}^T\boldsymbol{\varepsilon}_{\tau-1}\right] \quad . \tag{B.7}$$

The second term on the right can be found to be $-\varepsilon_{j,\tau-1}$ as in (B.6). For the first term,

$$\begin{aligned}\frac{\partial}{\partial V_{j,\tau-1}}\frac{1}{2}\boldsymbol{\varepsilon}_\tau^T\boldsymbol{\varepsilon}_\tau &= \boldsymbol{\varepsilon}_\tau^T\frac{\partial}{\partial V_{j,\tau-1}}\boldsymbol{\varepsilon}_\tau = \boldsymbol{\varepsilon}_\tau^T\mathbb{W}\frac{\partial}{\partial V_{j,\tau-1}}f(V_{\tau-1}) \\ &= f'(V_{j,\tau-1})\sum_{k=1}^{N}\varepsilon_{k,\tau}w_{kj} \quad .\end{aligned} \tag{B.8}$$

Substituting (B.8) and (B.6) into the expression for $B_{j,\tau-1}$ (B.7),

$$B_{j,\tau-1} = \varepsilon_{j,\tau-1} + f'(V_{j,\tau-1})\sum_{k=1}^{N}B_{k,\tau}w_{kj} \quad . \tag{B.9}$$

The general recursion for $B_{t,j}$ is (after reintroducing the indicator variable β),

$$B_{j,t} = \begin{cases}\varepsilon_{j,t}\beta_{j,t} & t = \tau \\ \varepsilon_{j,t}\beta_{j,t} + f'(V_{j,t})\sum_{k=1}^{N}B_{k,t+1}w_{kj} & 1 \le t \le \tau\end{cases} \quad . \tag{B.10}$$

Turning to the sparsity-enforcing term (again omitting β),

$$D_{j,t} = \frac{\partial}{\partial V_{j,t}}\left[\frac{\lambda}{2}\sum_{\rho=1}^{\tau}(V_t - \boldsymbol{\mu})^T(V_t - \boldsymbol{\mu})\right] \quad . \tag{B.11}$$

Following similarly to the derivation for B above, when $t = \tau$,

$$D_{j,\tau} = \frac{\partial}{\partial V_{j,\tau}} \left[\frac{\lambda}{2} (V_\tau - \boldsymbol{\mu})^T (V_\tau - \boldsymbol{\mu}) \right] = \lambda (V_\tau - \boldsymbol{\mu}) \frac{\partial}{\partial V_{j,\tau}} (V_\tau - \boldsymbol{\mu}) = \lambda (V_{j,\tau} - \mu) \ . \tag{B.12}$$

When $t < \tau$, we follow (B.7)-(B.9), and find the general recursion for $D_{j,t}$ (reintroducing β),

$$D_{j,t} = \begin{cases} \lambda(V_{j,t} - \mu)(1 - \beta_{j,t}) & t = \tau \\ \lambda(V_{j,t} - \mu)(1 - \beta_{j,t}) + f'(V_{j,t} - \mu) \sum_{k=1}^{N} D_{k,t+1} w_{kj} & 1 \leq t \leq \tau - 1 \end{cases} \ . \tag{B.13}$$

The recursions (B.10) and (B.13) are used in the final weight update,

$$\Delta w_{ji} = -\eta \sum_{t=1}^{\tau} (B_{j,t} + D_{j,t}) X_{i,t} \ , \tag{B.14}$$

where the activation function derivative is,

$$f'(v) = \frac{a_1 a_2 \exp\left(-a_2 v + a_3\right)}{\{1 + \exp(-a_2 v + a_3)\}^2} \ . \tag{B.15}$$

References

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

B. Apolloni, A. Bertoni, P. Campadelli, and D. de Falco. Asymmetric Boltzmann machines. *Biological Cybernetics*, 61:61–70, 1991.

F. Attneave. Informational aspects of visual perception. *Psychological Review*, 61:183–193, 1954.

D. Barber and P. Sollich. Gaussian fields for approximate inference in layered sigmoid belief networks. In *Advances in Neural Information Processing Systems 12 (NIPS*1999)*, Cambridge, MA, 2000. MIT Press.

H. B. Barlow. *The Mechanisation of Thought Processes*, chapter Sensory mechanisms, the reduction of redundancy, and intelligence, pages 535–539. Her Majesty's Stationery Office, London, 1959.

D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vols. I and II*. Athena Scientific, Belmont, MA, 1995.

D. Brook. On the distinction between the conditional probability and the joint probability approaches in the specification of nearest-neighbor systems. *Biometrika*, 51(3/4):481–483, December 1964.

E. M. Callaway. Feedforward, feedback and inhibitory connections in primary visual cortex. *Neural Networks*, 17:625–632, 2004.

Z. Chengxiang, C. Dasgupta, , and M. Singh. Retrieval properties of a Hopfield model with random asymmetric interactions. *Neural Computation*, 12:865–880, 2000.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

A. Crisanti and H. Sompolinsky. Dynamics of spins systems with randomly asymmetric bonds: Ising spins and Glauber dynamics. *Physical Review A*, 37(12):4865–4874, 1988.

Y. Ejima, S. Takahashi, H. Yamamoto, M. Fukunaga, C. Tanaka, T. Ebisu, and M. Umeda. Interindividual and interspecies variations of the extrastriate visual cortex. *Neuroreport*, 14(12):1579–1583, August 2003.

D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, online:1–31, July 2006.

D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3:194–200, 1991.

K. Fukushima. Restoring partly occluded patterns: a neural network model. *Neural Networks*, 18:33–43, 2005.

K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982.

C. C. Galland. The limitations of deterministic Boltzmann machine learning. *Network*, 4:355–379, 1993.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

J. Gill. *Generalized Linear Models: A Unified Approach*. Sage University Paper Series on Quantitative Applications in the Social Scienes, 07-134, Thousand Oaks, CA, 2001.

S. Grossberg. Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, 23:187–202, 1976.

H. Gutfreund. *Neural Networks and Spin Glasses*, chapter The Effect of Synaptic Asymmetry in Attractor Neural Networks, pages 49–66. World Scientific Publishing Co., Teaneck, NJ, 1990.

J. Hawkins and S. Blakeslee. *On Intelligence*. Times Books, New York, 2004.

R. Hecht-Nielsen. A theory of the cerebral cortex. In *Proceedings of the 1998 International Conference on Neural Information Processing (ICONIP'98)*, pages 1459–1464, October 1998. Kitakyushu, Japan.

J. A. Hertz, R. G. Palmer, and A. S. Krogh. *Introduction to the theory of neural computation*. Addison-Wesley, Redwood City, CA, 1991.

G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Phil. Trans. R. Soc. Lond. B*, 352:1177–1190, 1997.

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

G. E. Hinton and T. J. Sejnowski. Optimal perceptual inference. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 448–453, 1983.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

P. O. Hoyer and A. Hyvärinen. A multi-layer sparse coding network learns contour coding from natural images. *Vision Research*, 42(12):1593–1605, 2002.

D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, 148:574–591, October 1959.

O. Johnson. *Information Theory and the Central Limit Theorem*. Imperial College Press, London, 2004.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. *Learning in Graphical Models*, chapter An Introduction to Variational Methods for Graphical Models, pages 105–161. MIT Press, Cambridge, MA, 1998.

E. R. Kandel, J. H. Schwartz, and T. M. Jessel. *Principles of Neural Science*. McGraw-Hill, fourth edition, 2000.

H. J. Kappen and J. J. Spanjers. Mean field theory for asymmetric neural networks. *Physical Review E*, 61 (5):5658–5661, 2000.

S. M. Kay. *Fundamentals of Statistical Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 1993.

S. M. Kosslyn, W. L. Thompson, and N. M. Alpert. Neural systems shared by visual imagery and visual perception: A positron emission tomography study. *Neuroimage*, 6:320–334, 1997.

G. Kreiman, C. Koch, and I. Fried. Category-specific visual responses of single neurons in the human medial temproal lobe. *Nature Neuroscience*, 3(9):946–953, September 2000.

K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2):349–396, February 2003.

T. S. Lee and D. Mumford. Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7):1434–1448, 2003.

T. S. Lee, D. Mumford, R. Romero, and V. A. F. Lamme. The role of the primary visual cortex in higher level vision. *Vision Research*, 38:2429–2454, 1998.

M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2): 337–365, February 2000.

M. Mezard, G. Parisi, and M. A. Virasoro. *Spin Glass Theory and Beyond*. World Scientific, Teaneck, NJ, 1987.

V. B. Mountcastle. *The mindful brain*, pages 7–50. MIT Press, Cambridge, MA, 1978.

J. F. Murray. *Visual Recognition, Inference and Coding Using Learned Sparse Overcomplete Representations*. PhD thesis, University of California, San Diego, 2005.

J. F. Murray and K. Kreutz-Delgado. An improved FOCUSS-based learning algorithm for solving sparse linear inverse problems. In *Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 347–351, Pacific Grove, CA, November 2001. IEEE.

J. F. Murray and K. Kreutz-Delgado. Learning sparse overcomplete codes for images. *Journal of VLSI Signal Processing*, 45:97–110, 2006.

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

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vis. Res.*, 37:3311–3325, 1997.

B. A. Olshausen and D. J. Field. *23 Problems in Systems Neuroscience*, chapter What is the other 85% of V1 doing? Oxford University Press, Oxford, 2005.

G. Parisi. Asymmetric neural networks and the process of learning. *J. Phys. A: Math. Gen.*, 19:L675–680, 1986.

C. Peterson and J. R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1(5):995–1019, 1987.

R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435:1102–1107, June 2005.

R. P. N. Rao. An optimal estimation approach to visual perception and learning. *Vision Research*, 39(11): 1963–1989, June 1999.

R. P. N. Rao and D. H. Ballard. Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, 4:721–763, 1997.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

E. T. Rolls and T. Milward. A model of invariant object recognition in the visual system: Learning rules, activation functions, lateral inhibition, and information-based performance measures. *Neural Computation*, 12(11):2547–2572, November 2000.

L. K. Saul and M. I. Jordan. *Learning in Graphical Models*, chapter A mean field algorithm for unsupervised neural networks, pages 541–554. MIT Press, Cambridge, MA, 1998.

M. I. Sereno, A. M. Dale, J. B. Reppas, K. K. Kwong, J. W. Belliveau, T. J. Brady, B. R. Rosen, and R. B. H. Tootell. Borders of multiple visual areas in humans revealed by functional magnetic resonance imaging. *Science*, 268(5212):889–893, May 1995.

H. Sompolinsky. Statistical mechanics of neural networks. *Physics Today*, 41(21):70–80, December 1988.

C. F. Stevens. An evolutionary scaling law for the primate visual system and its basis in cortical function. *Nature*, 411:193–195, May 2001.

E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects and parts. In *International Conference on Computer Vision (ICCV 2005)*, volume 2, pages 1331–1338, 2005.

Y. W. Teh and G. E. Hinton. Rate-coded restricted boltzmann machines for face recognition. In *Advances in Neural Information Processing Systems 13 (NIPS2000)*, pages 908–914, Cambridge, MA, 2001. MIT Press.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.

S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6):520–522, June 1996.

W. E. Vinje and J. L. Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(18):1273–1276, February 2000.

M. Welling and Y. W. Teh. Approximate inference in Boltzmann machines. *Artificial Intelligence*, 143(1): 19–50, January 2003.

R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990.