Temporal Continuity Learning for Convolutional Deep Belief Networks

Carl Doersch CDOERSCH@ANDREW.CMU.EDU
Tai Sing Lee TAI@CNBC.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15289

Gary Huang GBHUANG@CS.UMASS.EDU
Erik Learned Miller ELM@CS.UMASS.EDU
University of Massachusetts Amherst, Amherst, MA 01003

Keywords: temporal continuity learning, unsupervised learning, computer vision

Abstract

The human visual system can robustly recognize objects, even though a single object can project many different images onto the retina. Furthermore, humans learn to perform this task from mostly unlabeled data. The goal of this work is to develop a computer algorithm which can replicate this sort of learning. One approach to this problem is called temporal continuity learning. This theory assumes that images close together in time are likely to contain the same object, and therefore that the visual system should learn representations that vary slowly in time. A different approach uses Deep Belief Networks. With DBNs, the goal of the learning is to maximize the likelihood of the training data in the marginal distribution of the Deep Belief Network. Interestingly, these approaches use entirely different heuristics to measure how 'good' a representation is. In this work, I hope to create an algorithm which uses both of these heuristics to form a better representation of images than either heuristic could produce on its own.

1. Introduction

Object recognition has proved a difficult task for computers, even though object recognition in humans is rapid and apparently effortless. The neural basis for this ability appears to reside in Inferotemporal cortex (IT), where neurons are sensitive to particular objects or patterns. Furthermore, each such IT neuron will respond to its preferred pattern even when the pattern is moved on the retina, or its pose is changed, or it is illuminated differently. The response patterns of these neurons are even more remarkable since they are learned from essentially unlabeled data in infants. How does the brain know which images correspond to the same object?

A proposal initially proposed by Hinton (Hinton, 1989) (p 208), and later called temporal continuity learning, solves the problem of deciding which images belong to the same object by assuming that images close together in time correspond to the same object. This leads to a straightforward intuition for a neural learning rule: if a neuron was active recently, then it should strengthen the connections to all neurons that it is currently receiving activation from. Földiák (1989) showed that this learning rule can be used to learn complex-cell connectivity fields when the input is simple-cell activations. Similar learning rules were later shown to perform more complex tasks, such as discriminating characters (Wallis & Rolls, 1997) and simple three-dimensional objects (Stringer & Rolls, 2002). Furthermore, temporal continuity learning has been demonstrated in human Inferotemporal Cortex (Li & DiCarlo, 2008; Wallis & Bülthoff, 2001).

It is only natural to ask whether an entire visual system can be learned with just a neural implementation of temporal continuity learning. Unfortunately, modern simulations which strive for biological plausibility, such as the Trace Learning framework (Földiák, 1989; Wallis & Rolls, 1997) are usually applied to relatively

Final submission for Carnegie Mellon University Senior Thesis.

simple, synthetic problems. Of the implementations of temporal continuity learning which have been created for the sake of computer vision, perhaps the most popular is Slow Feature Analysis (SFA) (Wiskott & Sejnowski, 2002). However, even SFA has a number of limitations: for example, in the classic implementation, computation time is $\mathcal{O}(N^4)$ where N is the number of pixels in the input. Perhaps more a fundamental limitation, however, is inherited from the temporal continuity framework itself: SFA cannot learn features that are not slow. Therefore, it struggles to learn features like edge-detectors, even though edge-detectors are present in the human visual system. Thus, Wiskott and Sejnowski (2002) hard-coded gabor filters into the first layer of their simulation.

There are many ways to learn edge-detectors from natural image data. Perhaps the most famous algorithm came from Olshausen and Field (1996), where it was shown that the optimal representations of images, under the constraint that the representation be sparse, involves units which have response properties similar to simple cells. In this case, "optimal" is in terms of the performance of an autoassociator: the learned representation was the one which minimized reconstruction error when the weights in the autoassociator were trained with backpropagation.

More recently, neural networks have begun to make use of techniques designed for graphical models. Notably, Deep Belief Networks (DBNs) (Hinton et al., 2006) have demonstrated good performance when recognizing handwritten digits. Furthermore, when they are constrained to be sparse, the units in a DBN will learn receptive fields similar to simple cells (Lee et al., 2008).

An important difference between DBN's and autoassociators is that each node in a DBN is probabilistic. Thus, sampling the states on one layer given other layers allows for some uncertainty. In particular, if we treat the DBN as a generative model, and if we assume that units in the deepest layers actually represent the presence or absence of objects and features in an image (which is the ideal representation of an image), then a DBN better reflects our intuitions about how images come about in the real world. That is, even after we know that an object is present in an image, it is still uncertain the exact image that will be generated. We can imagine generating the image hierarchically: starting with the knowledge that the object is present, we probabilistically generate its sub-features, and then the sub-features of those sub-features, until we reach edges and pixels. In an autoassociator, however, the generative process is entirely deterministic. Thus, we lose the fact that a single internal representation may correspond to multiple images.

The non-determinism of DBNs is particularly useful in this work because we hope to learn complex cell responses. In an autoassociator, units that behave like complex cells are difficult to learn because it is not clear what should be generated on the input layer when a complex cell unit is active. By definition, complex cells respond to multiple disjoint input patterns, but in an autoassociator they can generate only one of them. DBNs have not yet been shown to learn complex cell responses, but the probabilistic generative process should mean that units behaving like complex cells are at least possible; thus, complex cell repsonses are one of the goals of this work.

One difficulty with DBNs, however, is that they are slow and require many training images. In (Hinton et al., 2006), the network was trained on 60,000 images, and the images were only 28 by 28 pixels. For learning higher-order features, it is helpful to use more detailed images. Thus, we extend the Convolutional DBN (CDBN) framework (Lee et al., 2009). This framework makes sampling faster because conditional probabilities may be computed using a fast convolution operation. Furthermore, the network requires less training data because what is learned at one location in an image is propagated to all locations in the network.

Therefore, I chose the CDBN framework as a starting point for this work. The Temporal CDBN (TCDBN) described in the Methods section is a modification of a CDBN which allows information to propagate through time, thereby allowing temporal continuity learning in this framework.

2. Related work

The success of this work depends heavily on the hierarchical nature of images. That is, we assume that objects tend to be made up of parts, which are in turn made up of simple features that are based on edges and regions of color. This view of object recognition has a long history, starting with Fukushima's Neocognitron (Fukushima, 1980), which was designed for character recognition. The Neocognitron was successful because it acknowledged that the features of a character (for example, the T-junctions in the letter 'A') tend to be present in all images of the character, although the relative positions of the features may vary greatly. Thus, the Neocognitron recognized the features first, and then composed them into characters while remaining insensitive to the exact positions of

the features.

One notable descendant of the Neocognitron is the Convolutional Neural Network (CNN), which learns the best features at each layer of the visual hierarchy via backpropagation. These networks have been successful in many vision problems such as character recognition (LeCun et al., 1998), categorization of rigid objects (LeCun et al., 2004), and obstacle avoidance for robots (LeCun et al., 2005), which again confirms the utility of the hierarchical approach to object recognition. It is also worth noting that unsupervised pre-training may be applied to a CNN in a manner similar to the unsupervised pre-training usually used to train DBN's; doing so achieves modest improvements in performance (Ranzato et al., 2007).

A recent extension of the CNN which is of particular interest in this paper is proposed by Mobahi, Collobert, and Weston (2009). Their model extended the CNN framework to video data, making use of temporal continuity information by explicitly penalizing, in their objective function, differences between the representations of consecutive frames in a video. The goal is to learn representations that are invariant to the changes that happen over small time scales, making their model perhaps the most similar in spirit to this work out of all the papers that I have found in my survey. Their empirical success on standard object recognition databases provides further support for the use of temporal continuity data in deep architectures.

However, one criticism of purely feedforward architectures is that top-down hypotheses cannot be used to tune the representations at lower levels of the hierarchy during the inference process. Intuitively, a mechanism which can constrain the interpretations allowed in the lowest layers of the visual hierarchy would be useful because small, local image patches are inherently ambiguous (Oliva & Torralba, 2007). A Bayesian account of hierarchical image processing is appealing in this case, because such models easily allow top-down hypotheses to constrain low-level representations. Furthermore, hierarchical Bayesian inference can also explain physiological data from visual cortex (Lee & Mumford, 2003).

A number hierarchical Bayesian models of vision have been proposed, including the Deep Belief Networks on which the current model is based. Variants of Bayesian graphical models have been successful at a number of object recognition problems, in particular for deformable mbjects (Sudderth et al., 2005; Zhu et al., 2008). Bayesian graphical models for vision may also incorporate temporal continuity cues through the use of Markov Chains (Stepleton et al., 2009), although such work is still in its infancy. It is worth noting that complex graphical models such as these tend to lead to very difficult inference problems, especially as more constraints (like temporal continuity) are added.

Deep belief networks also fall into the category of hierarchical Bayesian models of vision, and there have been a number of algorithms that have used Deep Belief Networks with a temporal component. Notably, only minor modifications to the CDBN make it suitable for processing audio data. In a recent work, these CDBNs were shown to have state-of-the-art classification performance on a number of audio databases (Lee et al., 2009b). Conditional RBMs stacked into Deep Belief Networks have been used model human motion, and perform tasks like interpolating motion data (Taylor et al., 2007). It is important to note, however, that the goals of this paper differ from those of (Lee et al., 2009b) and (Taylor et al., 2007). These two papers model information that inherently contains a temporal component. That is, it is not possible to identify a speaker from a single audio sample, nor is it possible to identify a human motion from a single frame of motion-capture data. Thus, a DBN solution to either of these problems requires extending DBNs to use temporal data. However, in the present work we are concerned only with object recognition: we use the temporal continuity heuristic during training in order to improve a DBN's representations of, and classification performance on, still images. While it is possible that the Temporal CDBN proposed here would work on audio data or motion capture data, it is not expected that it would perform better than the DBNs specifically designed for this purpose. To my knowledge, there have not yet been any attempts to use temporal continuity information as a heuristic for constraining the learning in DBNs.

Because we are interested in DBNs for the way that they can select good features for representing input images, it is also worth taking a moment to examine other attempts to improve the representation power in networks that rely primarily on temporal continuity learning to derive their representations. In particular, we are interested in algorithms which were able to extract edge-like features, because edges have classically been very difficult to learn with only temporal continuity. There have been a small number of successful attempts at this problem. Hurri and Hyvärinen (2003)'s network learned Gabor filters using a learning rule related to SFA, although their objective function for SFA gave high scores to both features that were slow and features that oscillated rapidly from positive to negative activation values. Bergstra and Bengio's (2009) network learned units similar to Gabor filters

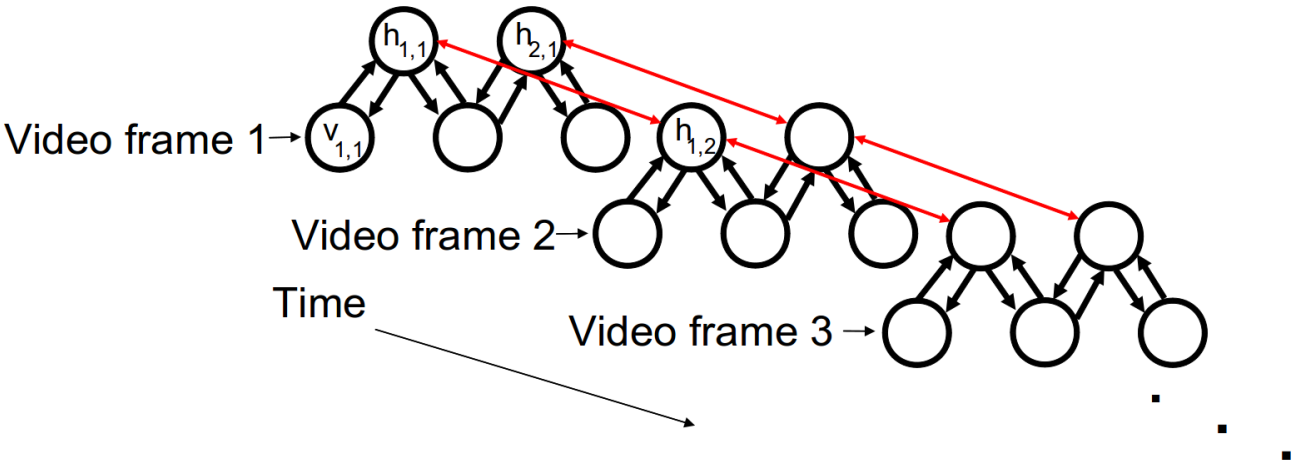

Figure 1. A toy temporal DBN for making the notation explicit. If implemented, this CDBN would operate on a video of 3-pixel, one-dimensional images (the real network used 2-d images with hundreds of pixels on a side)

only because they were learned at the same time as complex-cell units, and only the complex cell activations were used in the objective function. While these networks are somewhat successful, I believe that DBN learning provides a more principled way to add selectivity to neural receptive fields, both due to its statistical interpretation and its demonstrated successes with feature learning.

3. Methods

3.1. Basic structure

The model I propose combines deep belief networks with trace learning. The general idea of this model is to create a deep belief network where each unit is aware of its state during the previous time instant. Thus, the deep belief network can learn, by itself, to allocate units representing features which vary slowly through time. This would happen because more invariant units would be able to more accurately predict their previous states, and would thus be able to form a better model of video data.

To make the model more concrete, consider a standard CDBN that receives as input an n-by-n image. We can extend this model to a video with k frames by copying the network k times, and assigning each network to a frame in the video. Next we make the model's units aware of continuity data. For any unit u in the original single-image deep belief network, there is a corresponding set of its copies $\{u_1, ..., u_k\}$ in the model for video, one for each time $1, ..., k$. Each u_t is connected to u_{t-1}. Thus, messages traveling along these temporal connections will carry the same information that the trace conveyed in the trace learning model.

3.2. notation

I will describe the structure of the Temporal Continuity Convolutional Restricted Boltzmann Machine (one layer of a Deep Belief Network) shown in Figure 1. This network in the diagram is simplified in a few important ways relative to the network in the experiments. First, each input image in the diagram is one-dimensional; therefore, $v_{i,j}$ refers to the pixel in position i in frame j (Note: I use the words 'frame' and 'timestep' interchangeably). However, the original CDBN used 2-dimensional images that were on the order of hundreds of pixels on a side. Second, this network only has one 'group', whereas the original CDBN had 24 in the first hidden layer. Essentially, having 24 'groups' is like having 24 separate RBM's that are all attempting to explain the same image data. It is straightforward to generalize this procedure to multiple groups. Thus, in this explanation, each hidden unit will have two subscripts: $h_{i,j}$, where i is the position within the hidden layer, and j is the timestep. However, in the true network, each hidden group was 2-dimensional, sized such that there was one unit for each element in the 'valid' convolution between the input image and the group's weight matrix. Note that the convolution operation constrains the number of hidden units we may have in a block. Since the kernel size $N_W = 2$, and the number of visible units $N_V = 3$, we must have $N_H = N_V - N_W + 1 = 2$.

Thus, there are exactly three parameters in this toy network: w_1 which is the left weight (connecting $h_{1,j}$ to $v_{1,j}$ and $h_{2,j}$ to $v_{2,j}$), w_2 which is the right weight (connecting $h_{1,j}$ to $v_{2,j}$ and $h_{2,j}$ to $v_{3,j}$), and the temporal weight W_t, which connects $h_{i,t}$ to $h_{i,t+1}$ for all i, t. The temporal weights are shown in red. Let $N_T = 3$ be the number of timesteps.

3.3. Probability distribution

The probability distribution defined over this graph is essentially identical to that of (Lee et al., 2009) paper, except there is an extra term for the temporal weights. Therefore, we have:

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

Where E is the energy function:

$$\begin{aligned} E(v,h) = \quad & -\sum_{t=1}^{N_T}\sum_{i=1}^{N_H}\sum_{r=1}^{N_W} h_{i,t} W_r v_{i+r-1} \\ & -\sum_{t=1}^{N_T-1}\sum_{i=1}^{N_H} h_{i,t} t h_{i,t+1} \\ & -\sum_{t=1}^{N_T}\sum_{i=1}^{N_H} b h_{i,t} \\ & -\sum_{t=1}^{N_T}\sum_{i=1}^{N_V} c v_{i,t} \end{aligned} \tag{1}$$

And Z is a normalization constant that depends on the weights and biases:

$$Z = \sum_{v,h} exp(-E(v,h))$$

3.4. Weight updates

We wish to compute the derivative of the log probability of the data with respect to a weight $W_{i,j}$. To do this, we start by computing the derivative with respect to one connection only. In the end, we will compute the update for this parameter by summing up the updates for all connections it participates in.

The log likelihood may be written as:

$$\begin{aligned} L(v_0) = \quad & log(\textstyle\sum_h exp(-E(h,v_0))) \\ & -log(\textstyle\sum_{h,v} exp(-E(h,v))) \end{aligned}$$

Taking the derivative of this, we have:

$$\begin{aligned} \tfrac{\partial}{\partial W} L(v_0) = \quad & \frac{\frac{d}{dW}\sum_h exp(-E(h,v_0))}{\sum_h exp(-E(h,v_0))} \\ & -\frac{\frac{d}{dW}\sum_{h,v} exp(-E(h,v))}{\sum_{h,v} exp(-E(h,v))} \end{aligned}$$

$$= \quad \frac{\sum_h -\frac{\partial E(h,v_0)}{\partial W} exp(-E(h,v_0))}{\sum_h exp(-E(h,v_0))} - \frac{\sum_{h,v} -\frac{\partial E(h,v_0)}{\partial W} exp(-E(h,v))}{\sum_{h,v} exp(-E(h,v))}$$

If we distribute the denominator of each term into the the sum in the numerator, we see that both terms become expected values under two different distributions:

$$= -\sum_h \frac{\partial E(h,v_0)}{\partial W} p(h|v_0) + \sum_{h,v} \frac{\partial E(h,v)}{\partial W} p(v,h)$$

$$= \left\langle \frac{-\partial E(h,v_0)}{\partial W} \right\rangle_0 + \left\langle \frac{-\partial E(h,v)}{\partial W} \right\rangle_\infty$$

The derivative of the energy function with respect to a weight for a particular configuration is just -1 if both units that the weight connects are on in that configuration, and zero otherwise. Thus, if the weight connects units a and b, we have:

$$= \langle ab \rangle_0 + \langle ab \rangle_\infty$$

The contrastive divergence approximation approximates the right term with the distribution after one step of Gibbs sampling:

$$\approx \langle ab \rangle_0 + \langle ab \rangle_1 \tag{2}$$

Note that I did not use the conditional independence properties of the RBM anywhere in this derivation. The conditional independence is only useful because it means we can use sampling (or variational techniques) to compute these expected values in (2) efficiently. However, in the case of the Temporal CDBN, it is still possible to compute these expected values efficiently via sampling, as I describe here.

3.5. Sampling

In an ordinary RBM, it is easy to get samples from the posterior over the hidden units because the hidden units are all independent conditioned on an input vector. Thus, we get the following formula for the probability of h_i:

$$p(h_i = 1|v_0) = \frac{\sum_{h,h_i=1} \frac{exp(-E(h,v_0))}{Z}}{\sum_{h,h_i=1} \frac{exp(-E(h,v_0))}{Z} + \sum_{h,h_i=0} \frac{exp(-E(h,v_0))}{Z}}$$

Note that the Z's cancel. Furthermore, we can substitute in the energy function; some factoring leaves us with

$$= \frac{exp(-E_{h_i}(h_i = 1, v_0))}{exp(-E_{h_i}(h_i = 1, v_0)) + exp(-E_{h_i}(h_i = 0, v_0))} \tag{3}$$

Where E_{h_i} is h_i's contribution to the energy: all the terms from the energy function that depend on h_i. Note that the contribution $E_{h_i}(h_i = 0, v_0) = 0$, since every term in the sum is zero. Therefore, the final term in the denominator becomes 1. We can divide through by the numerator and get the standard sigmoid function:

$$= \frac{1}{1 + exp(E_{h_i}(h_i = 1, v_0))} \tag{4}$$

However, when we're using temporal continuity, the factoring in (3) doesn't work, because the units aren't independent. We must take a different approach.

I use the word 'chain' to refer to a set of hidden units that are all connected in time. Thus, the chain C_i is the set of all hidden units of the form $h_{i,t}$ for arbitrary t. As the units in any chain are not independent, they must be sampled together. We can compute the following formula for the probability of a chain being in a particular configuration:

$$p(C_i = c_i|v) = \frac{\sum_{h, C_i = c_i} \frac{exp(-E(h,v))}{Z}}{\sum_h \frac{exp(-E(h,v))}{Z}}$$

Again the Z's cancel, and we can factor out only those terms that depend on the state of c_i. What we are left with is in the standard form of an exponential family:

$$p(c_i|v) = \frac{1}{Z_{C_i}} exp(-E_{C_i}(c_i, v))$$

Where

$$\begin{aligned} E_{C_i}(c_i, v_i) = \quad & -\sum_{t=1}^{N_T} \sum_{r=1}^{N_W} h_{i,t} W_r v_{i+r-1} \\ & -\sum_{t=1}^{N_T - 1} h_{i,t} t h_{i,t+1} \\ & -\sum_{t=1}^{N_T} b h_{i,t} \end{aligned}$$

This can be seen as a markov random field, where there is a factor for each of the t nodes in c_i, and a factor for each pair $h_{i,t}, h_{i,t+1}$. Therefore, the graph is tree-structured, and so the sum-product algorithm (also known as belief propagation) can get us the marginals over every node in C_i. It is convenient to introduce the concept of direction when describing the sum-product algorithm. Say that $h_{i,0}$ is at the left end, and h_{i,N_T}) is at the right end. In my implementation of the sum-product algorithm, we begin at the left end. We compute the marginals based only on evidence coming from the visible units and from units to the left; for the leftmost unit this is nothing. Therefore, computing these pseudo-marginals reduces to computing the marginals of a unit that is independent of all other hidden units. The formula given in (4) may be used.

We proceed toward the right, each time computing the marginals on each unit given the bottom-up evidence and the evidence to its left. In practice, the evidence from below and evidence from the unit to the left are combined in a Bayesian way:

$$p_m(h_{i,t} = 1|p_m(h_{i,t-1}), v) = \frac{p_v(h_{i,t}=1)[(1-p_m(h_{i,t-1}))+p_m(h_{i,t-1})exp(W_t)]}{p_v(h_{i,t}=1)[(1-p_m(h_{i,t-1}))+p_m(h_{i,t-1})exp(W_t)]+p_v(h_{i,t}=0)} \tag{5}$$

Where p_v is the probability based only on the visible units, given in (4), and p_m are the marginal probabilites previously computed for the unit to the left (the 'messages' of the sum-product algorithm) . Note that the numerator is marginalizing over the probabilities of the unit to the left; the edge weight W_t only contributes to the probability when both $h_{i,t}$ and $h_{i,t-1}$ are set to 1.

According to the theory of the sum-product algorithm, the marginals at the right end of the chain are exact. Intutively, this is because there is no longer any evidence to the right that is being ignored. Thus, we can sample from h_{i,N_T} using these marginals. From here, sampling proceeds from right to left; each time a sample is taken from a hidden unit, the unit to its left must update its own marginals using a bayesian formula very similar to equation (5), except that $p_m(h_{i,t-1})$ is replaced with the sample from the unit to the right, and $p_v(h_{i,t} = 1)$ is replaced with the marginal probability computed on the right pass.

Of course, the sum-product algorithm can only be used to compute marginals; its use as a sampling mechanism was something I came up with. It is justified because this algorithm is equivalent to running the sum-product algorithm N_T times, each time to get a sample from the rightmost unit in the chain that has not yet been sampled. Each time we do this, the new sample on unit $h_{i,t}$ becomes a fixed input to the unit to its left, $h_{i,t-1}$. Thus, the next run of the sum-product algorithm can compute the exact posterior over $h_{i,t-1}$, since it will treat the sample on $h_{i,t}$ in much the same way it treats the values of the visible units.

3.6. Why is this an improvement over the CDBN?

Unfortunately, the reason here is an intuitive one, rather than a mathematical one. However, the reasoning is similar to the reason why the trace learning rule was an improvement over pure hebbian learning. The goal is to bias the network to favor bottom-up representations that vary slowly through time. I say 'bottom-up' because the goal is not that time-connections should greatly improve the descriminative performance through their effect during the descriminative process. The goal is that they should alter the inter-layer weights that the DBN learns.

Consider the problem of learning complex cell responses (this is the simlest case; one might similarly imagine units in higher layers learning rotation invariance). If we train on videos, the network will most likely learn edge-detectors (for the same reason the CDBN learns edge detectors). Then the time-connections will become positive, because an edge at orientation θ and position p at time t is a good predictor of an identical edge at time $t + 1$.

Say that one chain of hidden units is sensitive to orientation θ at position p. If this chain has a positive time-weight, then it will predict that any occurrence of an edge (θ, p) is likely to be preceeded and followed, temporally, by more edges (θ, p). However, this isn't the best model it could have of the world; in the true distribution of images, edges like (θ, p) are good predictors of edges like (θ, q) where q is a position close to p. The contrastive divergence learning rule will pick up on this: it will see that units in this chain are more likely to be active whenever lines like (θ, q) are shown in the chain's receptive field. Therefore, the learning rule will tend to modify the weights so that lines like (θ, q) will activate this chain. Over time, it is expected that this invariance will build up until the units behave like complex cells.

4. Graphics card implementation

The original implementation of this network, when training on images from the Kyoto dataset (Doi et al., 2003), required approximately one week of training time when running on an Intel Core 2 Duo processor clocked at 2GHz. This made the sort of iterative experimentation required for optimizing the network's hyperparameters prohibitively time-consuming. Therefore, I re-implemented the entire learning procedure to run on a graphics card, using the free Matlab library GPUmat (GPyou group, 2010). This re-implementation required approximately 40 hours and resulted in a speedup of approximately 5.5x, all with only minor reductions in code legibility.

5. Results

The goal of this work was twofold. The first was to improve the performance of the CDBN architecture on standard vision problems. The second goal was to show that the temporal modification of the CDBN increased the invariance the invariance of its representations. Unfortunately, however, it was not possible to fully explore the performance of the TCDBN relative to the CDBN, because I was never able to make the CDBN function as it was supposed to.

The results of training are shown in figure 3. Differences are clearly visible between the second-layer features that two networks learned. Most of the features that the original CDBN learned on each of its second layer groups are interpretable as remarkably clean combinations of the features from two first-layer groups, most often forming corner-detectors or boundary detectors. Our features, however, are either simple gabor filters, or much more complicated features that are difficult to interpret. In general, I found that increasing the weight decay, increasing the learning rate, and increasing the λ controlling sparsity tended to lead to simpler, cleaner features on the second layer; changing the parameters in the opposite diretion tended to result in features that are messier and in the worst cases degenerate. However, the values of the parameters also interacted heavily, making the results of simulations quite difficult to predict even for small changes in parameter values. Furthermore, I found that the results were quite sensitive to the initialization of the network. In general, it was very difficult to reproduce groups that appear to detect corners, although a small number of the groups shown in figure 3 do appear capable of detecting them.

At a first glance, the weights learned in the original CDBN work seem quite sensible, and appear to mirror V1 and V2 quite well. However, there are a number of features of the weights learned by Lee et al. (2009) that are somewhat surprising, and may point to differences between their implementation and ours. First of all, on the first layer, two of Lee et al. (2009)'s filters detect center-surround features rather than edges. In general, I had a difficult time replicating this; even in the final implementation, I could only learn center-surround units with a small probability, and was never able to produce a first layer that contained two of them. It is not entirely clear why the network should learn to detect such a feature; it is unlikely that such a feature would be sparse in natural images, nor does it

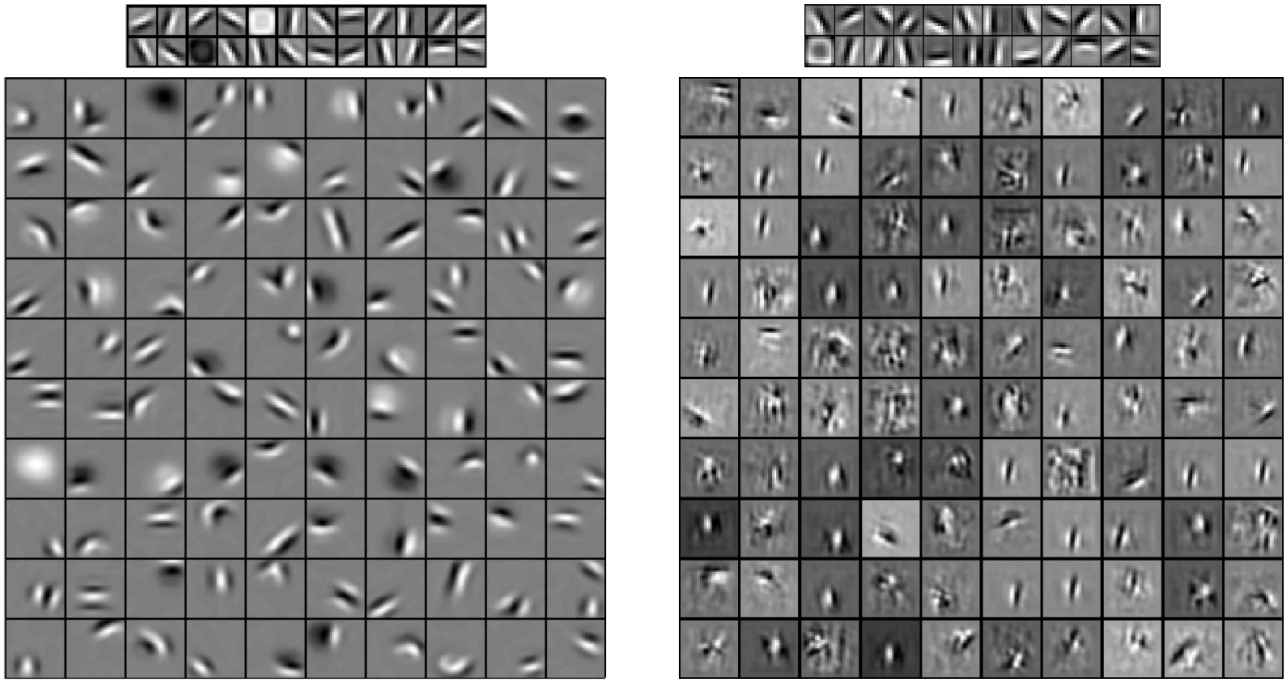

Figure 2. Visualizations of the weight vectors learned in a CDBN model, following the visualizations from (Lee et al., 2009). Left shows the weights learned in (Lee et al., 2009), right shows ours. Top shows the weights for the first layer of hidden units (if we think of the first hidden layer as 24 groups of feature detectors, then we may think of each group as detecting the feature depicted in one of the squares). The bottom shows a visualization of the features detected on the second hidden layer. The visualizations are constructed as a weighted linear combination of the first layer bases. The differences between the two sets of weights are discussed in the main text.

correspond particularly well with features encountered in the natural world. Moving to the second layer, we notice that again there are exactly two groups that detect large patches of uniform color. I was unable to replicate such feature detectors on my second layer, nor is it clear why they should appear on the second layer: the whitening step of the preprocessing undergone by each image should have removed almost all regions of uniform color from the image, Finally, there are a number of second-layer groups which appear to detect corners, but yet the two edges-detectors that make up the corner are not connected (see, for example, the fourth and eighth from the left in the top row). If such a seond-layer group g is actually corner-detector, and it detects corners by using edge features f and h that do not overlap spatially, then its activity should correlate well with those first-layer features in between f and h that have similar orientations to f and h, thus completing an image of the corner. Thus, we would expect g to fill in the gap between f and h by becoming sensitive to those features, as such a sensitivity would increase the probability of generating a correct image of a corner. Indeed, in my own simulations, each of the second layer units that learned non-degenerate weights became sensitive to a set of spatially overlapping first-layer features.

Much of my time this semester was spent attempting to replicate the results of (Lee et al., 2009), thus leaving relatively little time for actual testing. Unfortunately, I was never able to replicate these results, so I can only assume that this network is not equivalent to the one tested in the original CDBN paper, and so the applicability of the following results are limited.

5.1. Testing invariance with KL-divergence on artificial videos

I first attempted to show that temporal connections in a TCDBN actually encouraged invariance. In this experiment, I trained a TCDBN on artificial videos, and then tested whether the KL-divergences between the representations of nearby frames in the same video are reduced relative to the KL-divergences between frames in different videos. Specifically, I constructed two 12-frame videos out of 4 images from the Kyoto dataset. Letting the images be labeled A, B, C, and D,

Table 1. Matrix of pairwise KL-divergences between the posterior distributions over the hidden units given the different images. The entry with row labeled image i and column labeled image j shows the KL-divergence $KL(d(i)||d(j))$, where $d(x)$ is the posterior distribution over the second hidden layer of an ordinary CDBN given that the image x is input. This CDBN was trained using only the four testing images. These distributions were approximated using Mean Field. All values are $*10^6$.

	9	19	45	46
9	0	0.6342	0.6305	1.0613
19	1.2157	0	0.7008	1.2790
45	0.2293	0.3074	0	0.8203
46	3.5073	3.2780	3.4508	0

then the two videos are $\{AABBAABBAABB\}$ and $\{CCDDCCDDCCDD\}$. Separately, I trained a regular CDBN on the original four images. In the end, I compared the KL-divergences between all representations, to show whether the divergences between A and B and between C and D are lower in the TCDBN case than in the regular CDBN case. Showing this would lead to the conclusion that the temporal connections do indeed encourage invariant representations.

Tables 1 and 2 show the results of this invariance test. Table 1 shows the divergences between the representations of the different images in a regular CDBN; Table 2 shows the divergences in a TCDBN. To ease comparison between these two tables, Table 3 shows the ratio of each entry in 1 over the corresponding entry in 2.

Due to the nature of the KL-divergence measure, the rows in these tables are less variable in the columns; therefore, it is easier to see which distributions are closest to a given distribution by looking at the row for that distribution. In Table 3, we see that the representation of image 19 became more similar to the representations of all other images, but it gained by far the most similarity toward image 9, the image with which it shared a video. We see a similar story for image 45: its representation became less similar to the representations of all the other images, but the decrease in similarity was seen by far the least for image 46. For images 46 and 9, the results are less clear; the changes in the learned representations do not seem to be strongly affected by the image that 9 or 46 shared their videos with.

Unfortunately, it is also apparent that KL-divergence is a rather poor measure of the similarity between the representations of two different images. For example, the last row of table 1 makes it appear that the representation of image 46 is quite different from the other representations, but looking at the other rows this is much less clear. Overall, the KL-divergences are difficult to interpret. Therefore, the results of this experiment provide only weak evidence that the temporal weights can improve the representations.

Table 2. Matrix of pairwise KL-divergences between the posterior distributions over the hidden units given the different images, as above. In this case, posterior distributions were computed in a Temporal CDBN. The network was trained on two videos as described in the Evaluation secion. One video contaned images 9 and 19, and the other contained images 45 and 46. To allow a contribution from the temporal weights when computing the posterior distribution given an image, the input image was replicated 5 times to create a 5-frame video. The posterior distribution over the entire 5-frame CDBN was computed, but only the probabilities over the third frame were used to compute the KL-divergences. All values are $*10^6$.

	9	19	45	46
9	0	.8227	.8341	1,2427
19	.6474	0	.4586	.9007
45	.7529	.5999	0	1.0872
46	1.6245	1.3109	1.4977	0

Table 3. To ease qualitative comparison between the above two tables, the element-wise quotients of the tables are shown below. We hope to see large numbers in $(9, 19), (19, 9), (45, 46)$ and $(46, 45)$, and small numbers elsewhere.

	9	19	45	46
9	NaN	.7709	.7559	.8540
19	1.8778	NaN	1.5281	1.420
45	.3045	.5124	NaN	.7545
46	2.1590	2.5006	2.3040	NaN

5.2. Testing invariances with a standard invariance measure

Recently, a standard procedure has been proposed for measuring invariances in deep architectures (Goodfellow et al., 2009). In this procedure, we present video frames sequentially to the network and record the internal representations. We compute the measure by comparing the fraction of the time that a unit is active

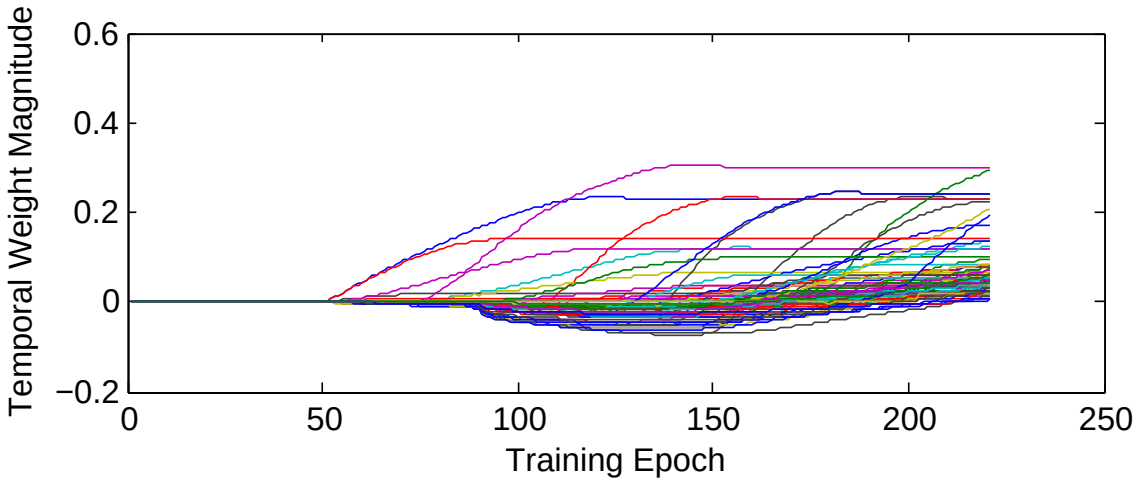

Figure 3. A plot of each of the 100 temporal weights as they learned over the course of 225 epochs. Note that all of the weights eventually learned positive values, indicating that the TCDBN learned features that were invariant over time, and that the network learned to predict the persistence of these features via its temporal weights.

given that it was active on a temporally close frame of the video (e.g. at any time within 5 frames of the current frame), versus the baseline fraction of the time that a unit is active. Note that this measure is only a relative measure: the measure is given with respect to a particular set of videos, and the procedure does not specify which set of videos to use. Furthermore, the procedure does not specify how to train the networks. Therefore, our results cannot be directly compared to those of Goodfellow et al. (2009). Instead, I performed the invariance measure procedure on both a CDBN and a TCDBN in order to compare their invariances.

The training and testing data was taken from the same library of videos used in (Goodfellow et al., 2009). Because I was most interested in the performance on natural image data and complicated transformations, I specifically used the 26 videos in which objects were rotated in 3D, in which the camera was translated, and in which other complex 3D motions were present, In this way, I ensured that the full complexity of the visual world was represented.

I used the first 10 frames of each video as training data. In the CDBN case, each frame was presented separately, whereas in the TCDBN case, each presentation consisted of 10 frames forming a coherent video sequence. The remainder of each video was used as testing data for the invariance measure.

To check whether the learning was proceeding as expected, I plotted the temporal weights as a function of the training epoch. Most importantly, we see that the temporal weights all learned positive values, which is consistent with the idea that the network had learned features that were somewhat invariant, and that it had begun using its temporal weights to predict a unit's activity on a given frame from the unit's activity on its temporally neighboring frames. Note that I fixed the timeweights at 0 until epoch 50; I found that if the timeweights were free before this point, they tended to behave as a surrogate for the hidden unit bias, and therefore became strongly negative early on in the simulation. This effect may still be seen even at epoch 50.

After testing, I found that the baseline invariance measure for the CDBN was 57.6548, while the invariance score for the TCDBN was 70.8295. Therefore, it is likely that the network has indeed learned a representation that is considerably more invariant than the original network when representing complex 3d transtormations.

6. Future work

It is clear that much remains to be done on this framework. The foremost concern is how to build CDBNs that can reliably learn V2-like detectors. Without such a baseline, it will be difficult to evaluate the effects

of alterations and extensions of the framework. In general, the DBN learning procedure is poorly understood, making it quite difficult to debug. Therefore, perhaps a stepping stone toward replicating the CDBN results would be to further investigate the learning procedure, either from a theoretical prospective or through simpler examples.

Once the CDBN is performing well, however, there are many questions that may be asked about the effect of temporal continuity information. It is natural to ask whether temporal continuity can improve performance on standard datasets, such as the Caltech101 database that the original CDBN was tested on. In theory, such work could have been attempted even without a properly functioning CDBN; however, time constraints prevented me from doing so.

We may also ask about links to vision in neural systems. At present, little is known about the mechanisms which allow neurons in V1 to learn simple and complex cell responses, and even less is known about the learning mechanisms in higher visual areas. However, if it can be shown that TCDBN response patterns correspond well with physiological data, then the principles behind DBN learning, such as Bayesian inference and maximum likelihood, may become a unifying framework for neural learning in the visual cortex.

References

Bergstra, J., & Bengio, Y. (2009). Slow, decorrelated features for pretraining complex cell-like networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams and A. Culotta (Eds.), *Advances in neural information processing systems 22*, 99–107.

Doi, E., Inui, T., Lee, T.-W., Wachtler, T., & Sejnowski, T. J. (2003). Spatiochromatic receptive field properties derived from information-theoretic analyses of cone mosaic responses to natural scenes. *Neural Computation*, *15*, 397–417.

Földiák, P. (1989). Learning invariance from transformation sequences. *Neural Computation*, *40*, 185–234.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetic*, *36*, 193–202.

Goodfellow, I., Le, Q., Saxe, A., & Ng, A. (2009). Measuring invariances in deep networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams and A. Culotta (Eds.), *Advances in neural information processing systems 22*, 646–654.

GPyou group, T. (2010). Gpumat: Gpu toolbox for matlab. `http://gp-you.org/`.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, *14*, 715–770.

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

Hurri, J., & Hyvärinen, A. (2003). Simple-cell-like receptive fields maximize temporal coherence in natural video. *Neural Computation*, *15*, 663–691.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

LeCun, Y., Huang, F.-J., & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of CVPR'04*. IEEE Press.

LeCun, Y., Muller, U., Ben, J., Cosatto, E., & Flepp, B. (2005). Off-road obstacle avoidance through end-to-end learning. *Advances in Neural Information Processing Systems (NIPS 2005)*. MIT Press.

Lee, H., Ekanadham, C., & Ng, A. (2008). Sparse deep belief net model for visual area v2. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*, 873–880. Cambridge, MA: MIT Press.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th International Conference on Machine Learning* (pp. 609–616). Montreal: Omnipress.

Lee, H., Pham, P., Largman, Y., & Ng, A. (2009b). Unsupervised feature learning for audio classification using convolutional deep belief networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams and A. Culotta (Eds.), *Advances in neural information processing systems 22*, 1096–1104.

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

Li, N., & DiCarlo, J. J. (2008). Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *Science*, *12*, 1502–1507.

Mobahi, H., Collobert, R., & Weston, J. (2009). Deep learning from temporal coherence in video. *Proceedings of the 26th International Conference on Machine Learning* (pp. 737–744). Montreal: Omnipress.

Oliva, A., & Torralba, A. (2007). The role of context in object recognition. *Trends in Cognitive Scheince*, *11*, 520–527.

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

Ranzato, M., Huang, F., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press.

Stepleton, T., Ghahramani, Z., Gordon, G., & Lee, T. S. (2009). The block diagonal infinite hidden markov model. *JMLR* (pp. 552–559).

Stringer, S. M., & Rolls, E. T. (2002). Invariant object recognition in the visual system with novel views of 3d objects. *Neural Computation*, *14*, 2585–2596.

Sudderth, E. B., Torralba, A. B., Freeman, W. T., & Willsky, A. S. (2005). Learning hierarchical models of scenes, objects, and parts. *ICCV* (pp. 1331–1338).

Taylor, G. W., Hinton, G. E., & Roweis, S. T. (2007). Modeling human motion using binary latent variables. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in neural information processing systems 19*, 1345–1352. Cambridge, MA: MIT Press.

Wallis, G., & Bülthoff, H. H. (2001). Effects of temporal association on recognition memory. *Proceedings of the National Academy of Sciences*, *98*, 4800–4804.

Wallis, G., & Rolls, E. T. (1997). Invariant face and object recognition in the visual system. *Progress in Neurobiology*, *51*, 167–194.

Wiskott, L., & Sejnowski, T. J. (2002). Slow feature analysis:unsupervised learning of invariances. *Neural Computation*, *14*, 715–770.

Zhu, L., Lin, C., Huang, H., Chen, Y., & Yuille, A. L. (2008). Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. *ECCV (2)* (pp. 759–773).