Temporal Continuity Learning for Convolutional Deep Belief Networks

Carl Doersch CDOERSCH@ANDREW.CMU.EDU
Carnegie Mellon University, Pittsburgh PA 15289

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 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

Preliminary work. Extended abstract for Carnegie Mellon University Senior Thesis.

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.

2. Related Work

There have been a number of algorithms using 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 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.

There have also been attempts to extend SFA to allow it to better represent such features as edges, which are less slow relative to the sorts of features SFA usually extracts. Hurri and Hyvärinen (2003)'s network learned Gabor filters using a learning rule related to

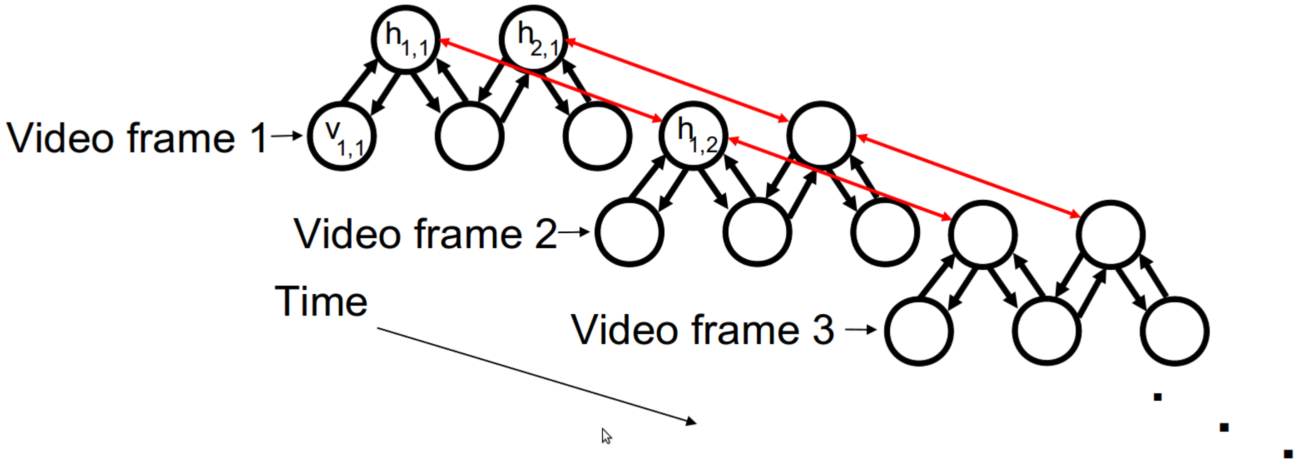

Figure 1. A toy temporal DBN for making the notation explicit. If implemented, this CDBN would operate on a video of 3-pixel, one-dimensional images (the real network used 2-d images with hundreds of pixels on a side)

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 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:

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

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:

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

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

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

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:

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

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

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

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:

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

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. Evaluation and Expected Conclusions

The first goal of this research is design a state-of-the-art computer vision algorithm. I plan to test this network using the same benchmark as the original CDBN algorithm: the Caltech 101 database. If the Temporal CDBN is indeed learning a better representation of the input, then the performance on this database should improve. Similar to (Lee et al., 2009), I will train the Temporal CDBN on natural videos that were unrelated to the Caltech 101 task.

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

As a stepping stone to this task, I will first show that the network is learning invariant representations. Perhaps the simplest test is to train a Temporal CDBN on videos, and then test 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 have constructed two 12-frame videos out of 4 images from the Kyoto dataset. Letting the images be labeled A, B, C, and D, 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 temporal CDBN case than in the regular CDBN case. Showing this would lead to the conclusion that the temporal connections do indeed encourage invariant representations.

There are other tests of invariance as well; notably, (Goodfellow et al.,) proposed an invariance measure which involves showing video sequences to to DBNs and measuring the differences between the representations of different frames in the videos. Thus, that work provides a standard way to measure directly the quantity we are trying to measure.

The second goal of this research is to present the Tem-

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

poral CDBN as a model of vision in humans. Showing that the representations in deep layers are invariant is a good first step. (Lee et al., 2009) showed that DBNs can learn simple-cell-like receptive fields; however, their use of pooling to create complex cells was not realistic and did not involve learning. I hope to show that the network can learn units which behave like complex cells. Inspection of the weights, along with simulations following the Hubel and Wiesel (1962) experiments, should be enough to establish whether such fields are present.

5. Results

Tables 1 and 2 show the results of the first invariance test proposed in the previous section. Table 1 shows the divergences between the representations of the different images in a regular CDBN; Table 2 shows the divergences in a temporal CDBN. 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. It is hoped that repetitions of these experiments, or the use of the techniques in (Goodfellow et al.,), will provide more interpretable results.

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.

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

Goodfellow, I., Le, Q., Saxe, A., & Ng, A. Measuring invariances in deep networks. In *Advances in neural information processing systems 22.*

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.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol*, *160*, 106–154.

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

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.

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

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.

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.

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.