Learning Perceptual Invariances: A Spatial Model

Stephen Eglen*, Jim Stone†, Harry Barrow*

*School of Cognitive and Computing Sciences,
University of Sussex,
Brighton BN1 9QH, England.
stephene,harryb@cogs.susx.ac.uk

†AI Vision Research Unit, Psychology Building,
Sheffield University, Western Bank,
Sheffield, S10 2TN, England.
stone@aivru.sheffield.ac.uk

Cognitive Science Research Paper 404

January 1996

Abstract

A novel unsupervised learning method for extracting spatio-temporal invariances has been developed in (Stone, 1995). The learning method works by trying to jointly minimise the short term variance of a unit's output, whilst maximising the long term variance of the output. The learning method has been applied to extracting disparity from a temporal sequence of random dot stereograms.

This paper reports on developing and applying the learning method to a spatial task, both in one and two dimensions. Random dot stereograms were used as input to a three layer feedforward network. After learning, output units in the network became selective for disparity. This confirms the usefulness of the learning method, and leads the way to creating a full spatio-temporal model, using both temporal and spatial information.

1 Introduction

An unsupervised learning method of extracting perceptual invariances from a sequence of images has been developed in (Stone, 1995). The method rests on the assumption that whilst pixel values over a patch of image can change quite rapidly, the underlying parameters (such as depth or disparity) usually vary smoothly. Stone and Bray (1995) have also applied the learning method to the problem of converting a place coding amongst a group of units into a value coding from one unit.

For a unit to be selective to one of these underlying perceptual invariances, such as depth, we should expect that its output will change smoothly over time in accordance with the invariance. Hence, the unit should have a low short term temporal variance. However, the unit can have a very low short term temporal variance (zero) if its output is always the same, regardless of its input. Hence, another constraint that can be placed on the unit is that it should have a large long term variance: over a long period of time, the unit's output should vary quite a lot. The learning method presented in (Stone, 1995) puts both these constraints into one measure F:

$$F = \log \frac{V}{U}$$

where V is the long term variance of the unit, and U is the short term variance. By maximising F, we will jointly maximise the long term variance of the unit's output and minimise its short term variance. (The log function is used so that the derivative of F is easy to compute - see Appendix A.)

This learning method was used by Stone in a feedforward network presented with a sequence of one dimensional random dot stereograms. Each input was a pair of random dot stereograms, with the disparity between the two images varying in a sinusoidal fashion over time. The learning method was used to maximise F for the one output unit. After learning, the output of the network was highly correlated ($r > 0.97$) with the disparity between the images.

The work presented here uses the same learning method to a similar problem in the spatial domain. Rather than having a sequence of images presented to the network over time, we have one large image where the disparity varies smoothly over the image. Additionally, instead of having one output unit, we now have an array of output units, and the learning rule is now applied to maximise F over all of the output units.

For brevity, the temporal model described by (Stone, 1995) will be called the 'temporal model', whereas the model presented here will be called the 'spatial model'.

1.1 Related Work

The spatial task used here is the same as the task used in (Becker, 1992). Using the IMAX method, Becker showed how maximising the mutual information between units of adjacent networks could be used to learn stereo disparity. However, using an architecture similar to that described here, pre-learning of the hidden layer was required in order for the system to learn disparity. This pre-learning consisted of maximising the mutual information between corresponding hidden layer units in adjacent networks. Maximising mutual information between output units *without pre-learning* was effective only if the hidden layer of each network was treated as a number of distinct clusters, with weight sharing between corresponding clusters in different networks. The need for these added constraints suggest that the merit function used in IMAX may not be best suited to extracting disparity.

2 Method

2.1 Inputs

Two large random dot stereograms were created, such that the disparity varied smoothly (either sinusoidally or in a gaussian manner) over the image. The images were also blurred with a gaussian filter, and normalised to have zero mean and unit variance. Both one and two dimensional images were created, and are described in more detail later.

2.2 Network Architecture

A three layer fully connected feedforward network was created, similar to the one used in the temporal model. To simplify the task, a shared weight system was used, as shown in Figure 1. The actual size of the shared network varied in experiments, but there was always just one output unit from the shared network. The number of output units in the virtual array was the same as the number of pairs of image patches.

The two input images were broken up into non overlapping patches. Matching patches from each image were copied into the input layer of the network, and the activity propagated to the output layer of the shared network. The output of the shared network was then copied into the appropriate part of the array of virtual output units.

For example, let the two input images be 500×1 pixels, and the input layer contain 5×2 input units. Both input images are therefore broken up into 100 patches of size 5×1. Patch n from both eyes is presented together as input to the network. The network computes the output of the network, and this output is then copied into unit n of the array of virtual output units. This procedure is repeated for all 100 patches, and corresponds to the network being swept across the two large images.

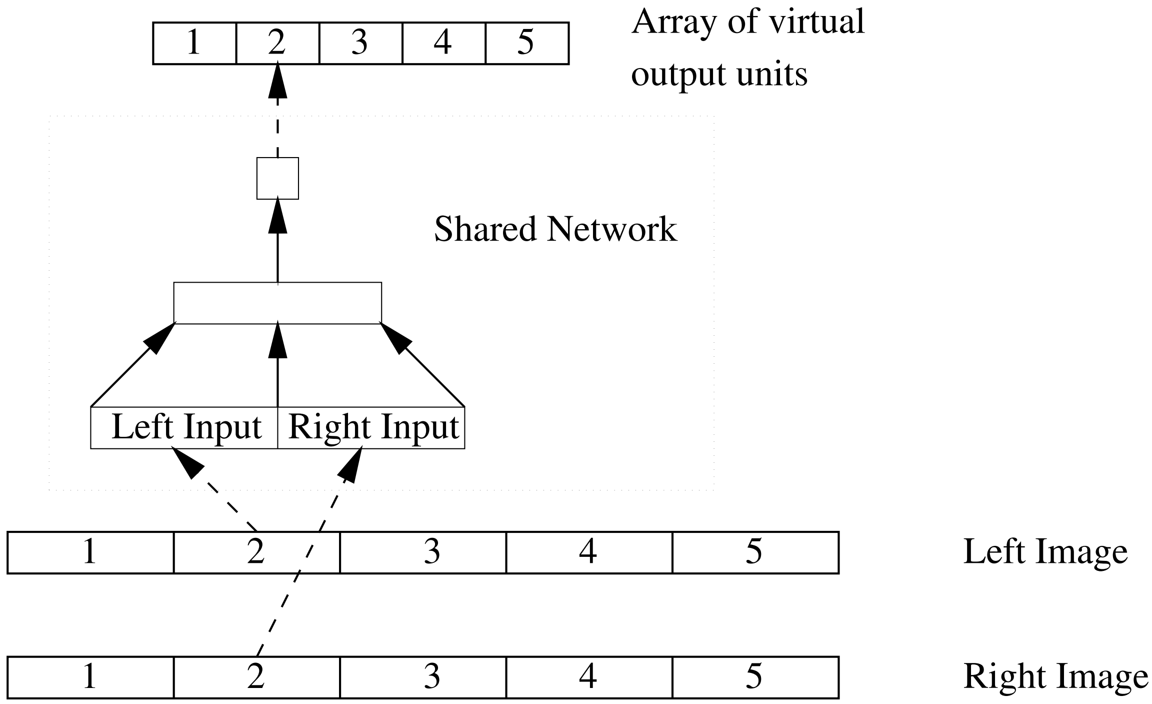

Figure 1: Architecture of the shared network. In this example configuration, the two images are broken down into 5 patches, and currently patch two is being presented to the network. The output of the shared network is then copied into the second virtual output unit.

2.2.1 Network Configurations

Two main configurations were used in this work: one dimensional and two dimensional. In the one dimensional configuration, input images were one dimensional and the virtual output units were arranged in a one dimensional array. In the two dimensional configuration, we used two dimensional input images and the virtual output units were arranged into a two dimensional array. The one dimensional version corresponds to a spatial analogy of the temporal learning task, whereas the two dimensional version is a novel extension.

In both configurations, units in each layer were fully connected to units in the next layer. Additionally, hidden layer units received an extra bias input. Input and output units used the identity transfer function for calculating the output of units, whereas the hidden units used a tanh transfer function. Appendix A.2 gives the equations describing network activity.

2.3 Learning Rule

The temporal model learning rule has been slightly adapted for the spatial model, although it essentially remains the same. In the temporal model,there is one output unit. The long term variance (V) and the short term variance (U) of the unit is measured as:

$$V = \frac{1}{2}\sum_{t=1}^{T}(\bar{z}_t - z)^2$$
$$U = \frac{1}{2}\sum_{t=1}^{T}(\tilde{z}_t - z)^2$$

where z is the output of the unit, $\bar{z}$ is the long term weighted average of the output unit, and $\tilde{z}$ is the short term weighted average of the output unit.

In this model, there is no temporal component, but instead there is an array of output units. Hence, the long and short term variances are measured as the long and short range spatial variances:

$$V = \frac{1}{2}\sum_{k}(\bar{z}_k - z_k)^2$$
$$U = \frac{1}{2}\sum_{k}(\tilde{z}_k - z_k)^2$$

where k ranges over all output units in the virtual output array. $\bar{z}_k$ and $\tilde{z}_k$ now represent the long and short range spatial variance of virtual output unit k. Therefore, the only differences in the spatial learning rule and the temporal learning rule is the way in which U and V are calculated, and in turn, the way in which the short and long term weighted averages are calculated. (Appendix A.3 show how U and V are calculated.)

2.3.1 Learning Paradigm

Conjugate gradient descent (Williams, 1991) was used to maximised F with respect to each weight in the network. Each line search of the conjugate gradient required evaluation of F and $\frac{\partial F}{\partial w}$ at several points in the weight space. After each epoch (defined here as one iteration of conjugate gradient descent), the correlation between network outputs and disparity was computed. Learning typically continued until the values of F and r stabilised. Full details of computing F and $\frac{\partial F}{\partial w}$ are given in Appendix A.

It is worth noting that this is an unsupervised learning method. The known disparity between input images is never used by the learning algorithm – we only use it for computing the correlation between the network output and disparity.

2.4 Free Parameters

The only free parameters of the model needed were those controlling the extent of spatial averaging for computing $\bar{z}$ and $\tilde{z}$. In this model, we used exponentially weighted kernels to compute these terms (see Appendix A.5), and so the kernel half lives (denoted by h_u and h_v) controlled the extent of spatial averaging. The results from the temporal model indicated that the half life h_v should be large enough such that $\bar{z}$ was a good approximation to the mean value of all of the outputs, whereas h_u should be set so that $\tilde{z}$ is averaging amongst something like ten percent of the output units. The size of the network was initially chosen to be similar to the network used in the temporal model – ten input units (five units for each image), between three and ten hidden units and one output unit.

2.5 Testing Network Performance

After learning, the weights were then frozen so that the network could be tested on other images not seen during learning. To do this, the new images were cut up into patches of the same size as those used during learning and presented to the shared network one at a time. The correlation r between the output of the network and the known disparity for the image patches was then measured. (The size of the test images did not need to be the same size as the images used in learning, since the virtual array is not used during testing.)

3 Results

Two sets of experiments were performed. The first set of experiments used one dimensional input images with the virtual array of output units arranged in a one dimensional array. The one dimensional configuration therefore was a spatial analogy of the temporal learning task described by (Stone, 1995). The second set of experiments used two dimensional input images with the virtual array of output units arranged in a (square) grid, and were the more interesting because of the extra dimension involved.

3.1 One Dimensional Network

Two input images of size 5000×1 pixels were generated to make 1000 pairs of 5×1 image patches. The disparity between a pair of image patches was set to some value between ± 1, which varied sinusoidally with a period of 1000 over the 1000 patches. These image patches were presented to a network with 10 $(2 \times 5 \times 1)$ input units, 10 hidden units and one output unit. The half lives were set in accordance with the half lives used in the temporal model: h_u was 32 and h_v was 3200. The virtual output array used here was a circular 1D array of 1000 output units.

Figure 2 shows the results after learning for 160 epochs. The final correlation between disparity and network output reached 0.937. Figure 3 shows how the network output developed during learning.

After learning, the network was then tested on two random dot stereogram images of size 1000×1, where this time the disparity varied sinusoidally between ± 1 with a period of 40. As seen from Figure 4, the network generalised well to this image pair, producing a correlation of 0.940 between disparity and output.

Similar experiments were performed, using inputs whose disparity varied sinusoidally with periods much less than 1000 for learning. In these experiments, the network would produce outputs highly correlated with disparity, but only if the short term half life, h_u was considerably reduced. For example, when the network was using the data set whose disparity varied sinusoidally over a period of only 40 patches as input during learning, the half life h_u had to be reduced to either 2 or 3 for the network to learn the disparity. One side effect, as noted by Stone was that reducing the short range half life increased the learning time (see Figure 4 of Stone, 1994).

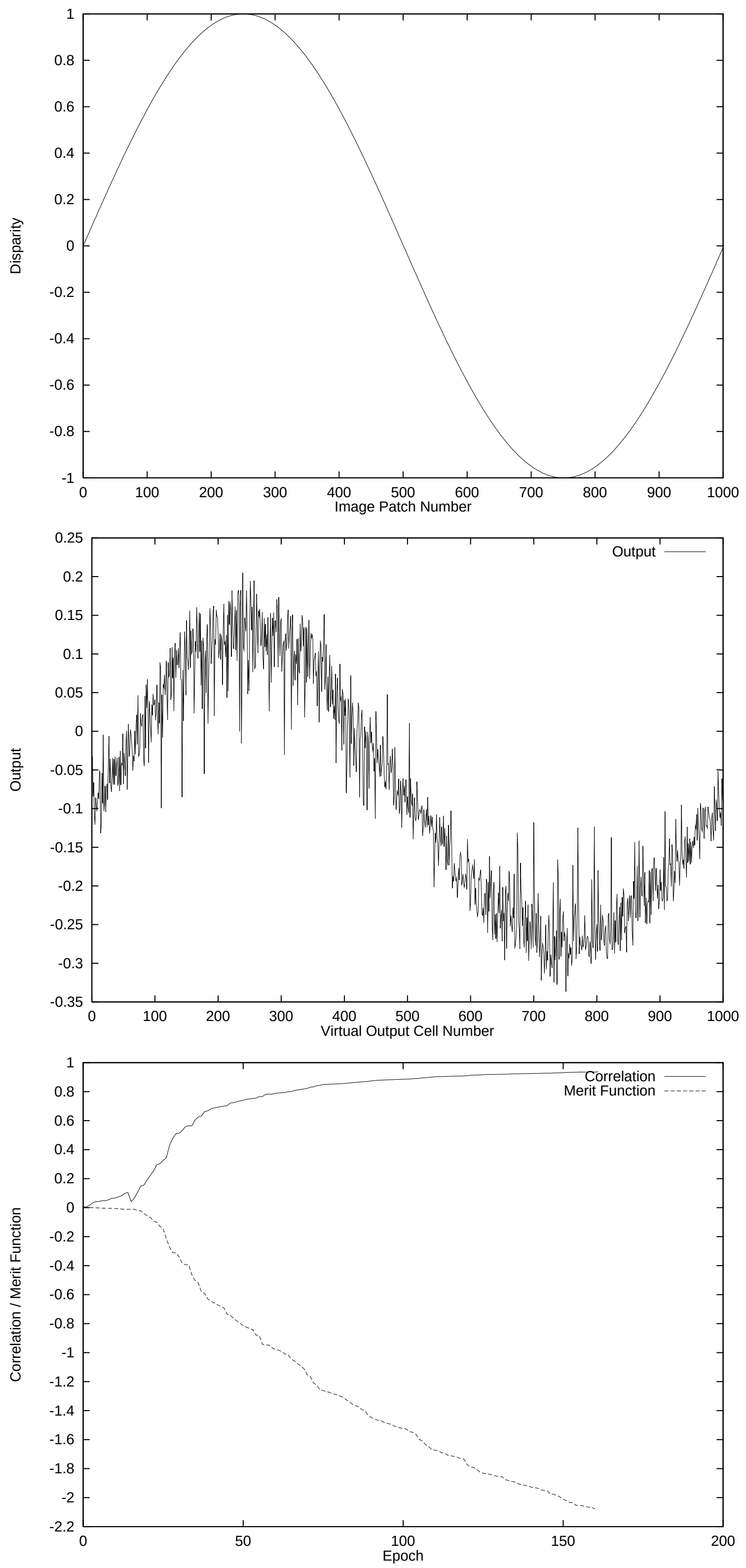

Figure 2: Results of the 1D network learning with the 1d random dot stereograms, where the disparity varied sinusoidally between ± 1 with a period of 1000 over the 1000 image patches. Top: Disparity values for the 1000 pairs of image patches used during learning. Center: Output from the network after learning. Bottom: Plot of correlation and merit function during learning.

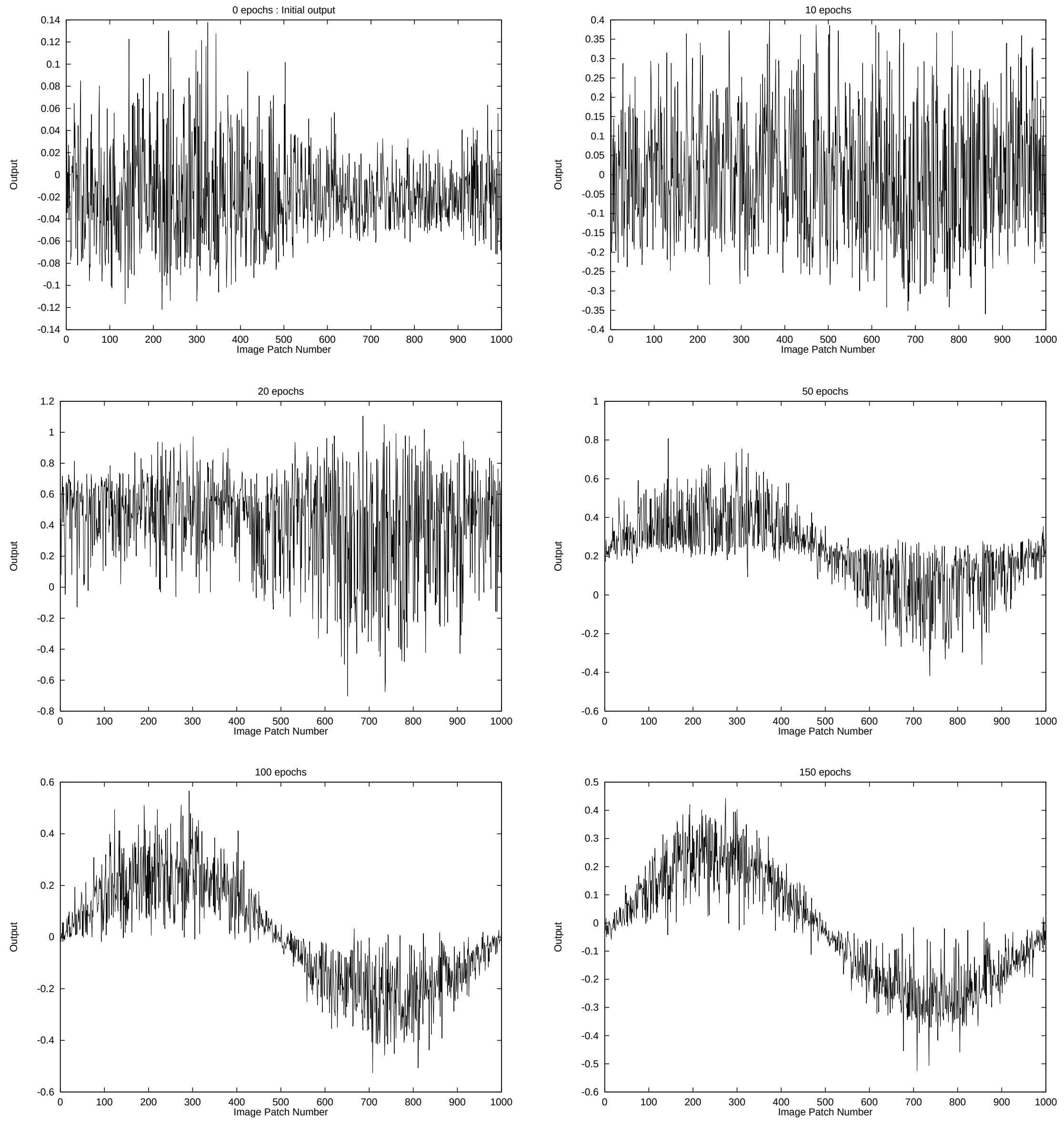

Figure 3: Development of network output during learning, using the 1D data described in Figure 2. Network outputs are shown left to right and top to bottom after 0, 10, 20, 50, 100 and 150 epochs.

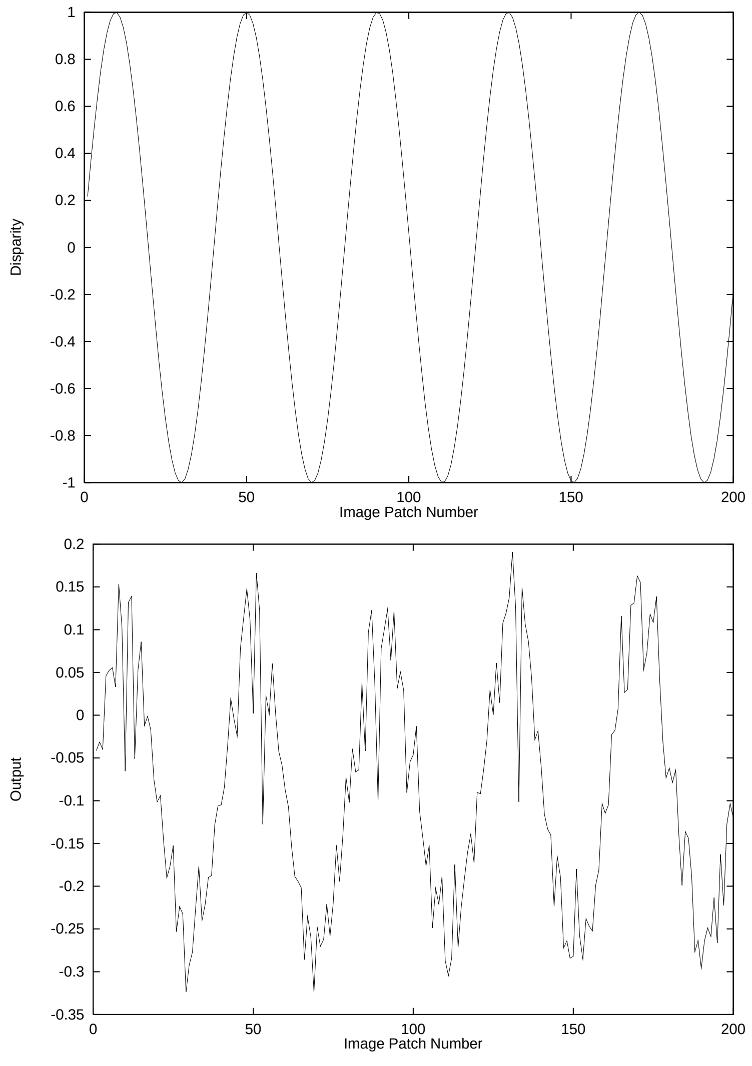

Figure 4: Results of testing the 1D network on 200 image patches unseen during learning. Input image patches were taken from a pair of random dot stereograms with disparity varying sinusoidally between ± 1 over a period of 40 patches. Top: Disparity values for the 200 pairs of image patches used for testing. Bottom: Output from the network.

3.2 Two Dimensional Network

A pair of random dot stereograms of size 120×120 pixels were created, such that the disparity between patches of the images was a gaussian function of the distance of the patch from the centre of the image (see Figure 5). These two images were then broken down into 3×3 non overlapping image patches. The shared network had 18 ($2 \times 3 \times 3$) input units, 5 hidden units, and 1 output unit, feeding into a virtual array of 1600 (40×40) output units. Figure 5 shows the network output after 1000 epochs, along with a plot of the merit function and correlation during learning. Figure 6 shows how the outputs develop during learning. As can be seen from Figures 5 and 6, the network has essentially learnt the disparity after 200 epochs, and the remaining 800 epochs are spent gradually improving the merit function.

(Note: The array of 40×40 disparity values and 40×40 output values are visualised as 40×40 greyscale images, with each disparity or output value represented by the pixel intensity: the larger the value, the brighter the pixel.)

3.2.1 Testing Network Performance

After learning, the weights were frozen and the network tested on a new random dot stereogram pair, where the disparity varied sinusoidally between ± 1 independently in both the horizontal and vertical directions of the images, as shown in Figure 7. 2500 (50×50) pairs of image patches were extracted from this new image pair and presented to the network. The correlation between network output (shown in Figure 7) and disparity was 0.890.

3.2.2 Hidden Units

We have not yet considered in detail the role of the hidden units. In some preliminary experiments, we modified the number of input units and hidden units in the network. Results from three of these networks are presented for comparison in Table 1. Clearly, the network with the most hidden units (15) performed the best when the 2D gaussian disparity images were used as inputs during learning. It also produced the best correlation on the test data (the 2D sinusoidal disparity images). However, much more work needs to be done looking at the hidden units, including looking at their receptive fields, to understand the role of the hidden units.

	After Learning		Testing
Network Size (Input:Hidden:Output)	Merit function	Correlation	Correlation
$(2 \times 3 \times 3) : 5 : 1$	-1.725	0.923	0.890
$(2 \times 5 \times 5) : 3 : 1$	-1.267	0.864	0.785
$(2 \times 5 \times 5) : 15 : 1$	-2.982	-0.972	-0.908

Table 1: Comparison of different network sizes. The 2D gaussian data was used during learning, and the 2D sinusoidal data for testing.

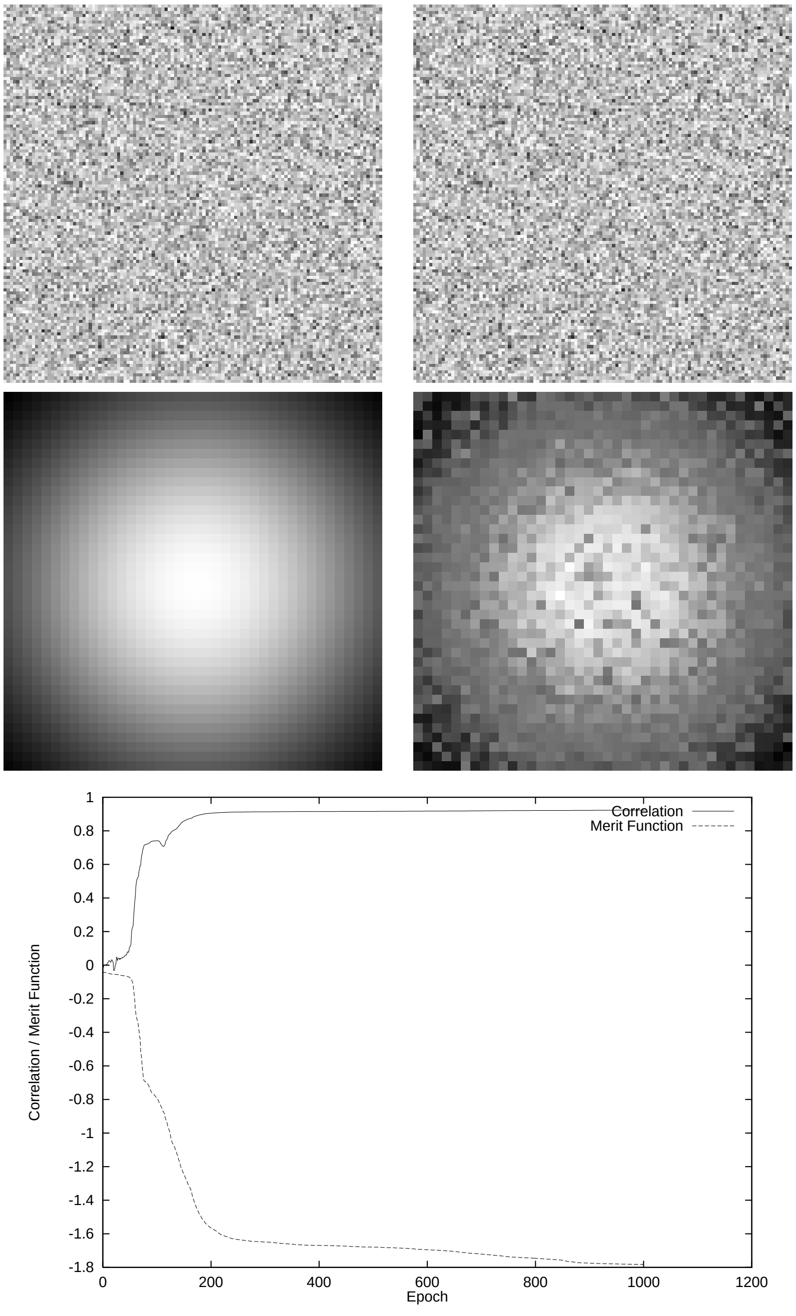

Figure 5: Results of the 2D network learning, using as input a random dot stereogram pair where the disparity between image patches decreased in a gaussian manner from the centre of the image. Top row: pair of random dot stereograms, used as input to the network. Centre left: Disparity between the two input images. Centre right: Output from the network after 1000 epochs of learning. Bottom: Plot of correlation and merit function as a function of learning time.

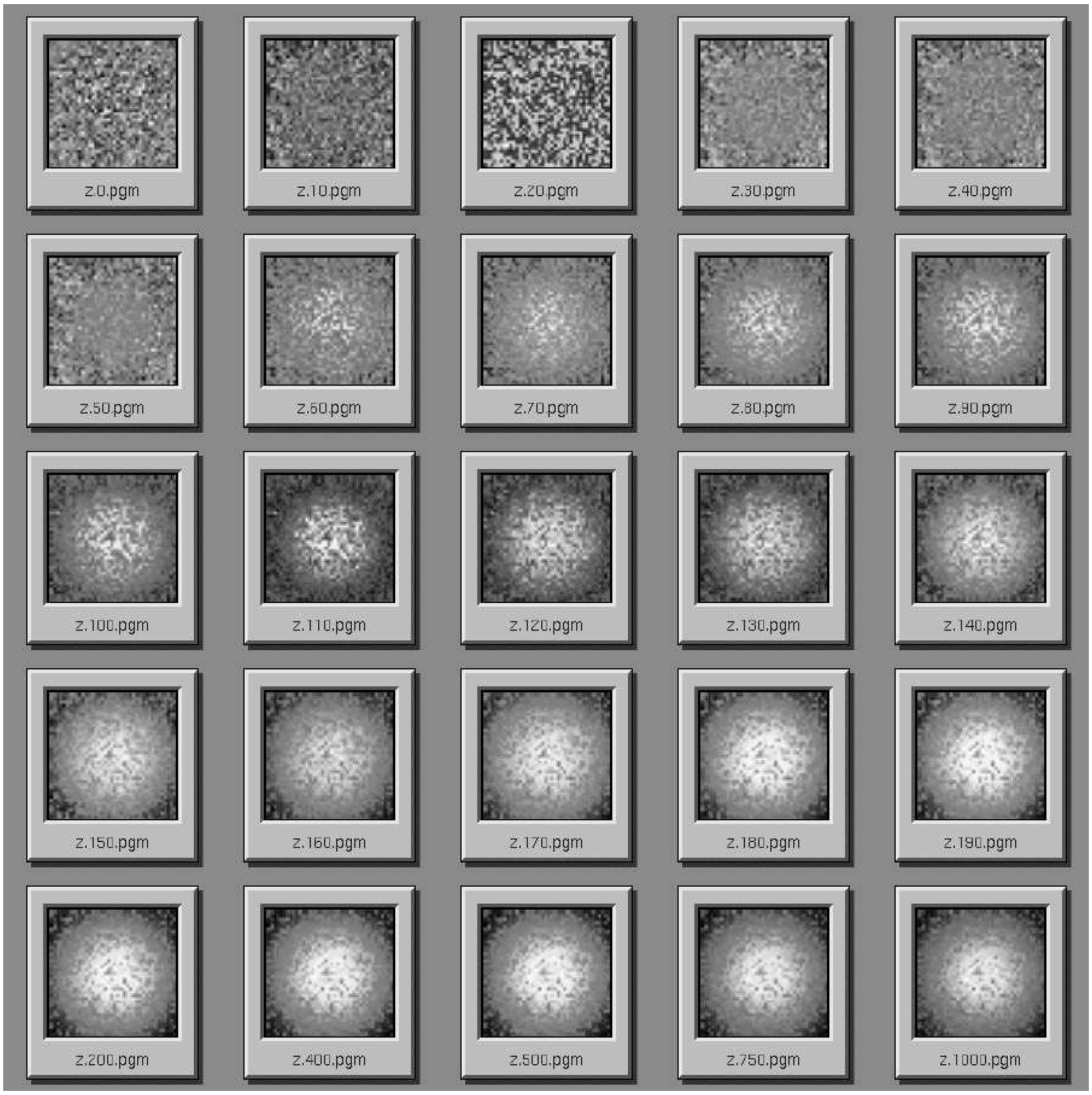

Figure 6: Evolution of the outputs during learning for the 2D network, using the data set presented in Figure 5 as input. The images, from left to right and top to bottom show network outputs after 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170,180, 190, 200, 400, 500, 750 and 1000 epochs.

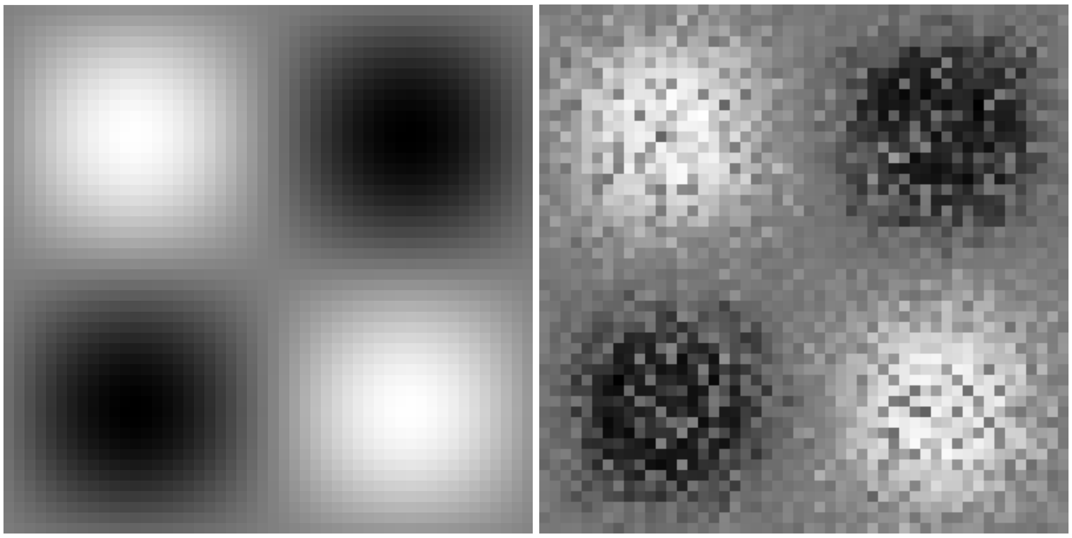

Figure 7: Results of testing the 2D network on a novel image pair. Left: Array of disparity values between the novel image pair. Right: Array of network outputs. The correlation between the disparity and the output is 0.890.

4 Conclusions

We have shown that the unsupervised learning model presented by (Stone & Bray, 1995) can be used in the spatial domain to learn disparity, in both one and two dimensions. This confirms the usefulness of the learning method.

This work can be developed in several directions. Firstly, it could be extended to be a non weight sharing network. This is a harder learning task than the shared network presented here, and will probably mean that more image patches will need presenting to the network. However, we expect that a non shared network should produce similar results to the ones presented here.

However, a more significant direction for this work is to apply the learning method to a full spatiotemporal task. This would effectively combine the temporal learning rule used in the temporal model and the spatial learning rule presented here. Input to the network would be a sequence of images. After presenting one input (or pairs of images in a stereo task) to the whole network, the spatial learning rule could be applied to the array of output units. Simultaneously, the temporal learning rule could be applied separately to each output unit. This would produce two error vectors for maximising the merit function with respect to the network weights.

A Mathematical Details of the Model and Learning Rule

This appendix provides the necessary mathematical details of the model.

A.1 Notation

Term	Meaning
x	total input to a unit
z	Output of a unit
$\bar{z}$	Long range weighted average of z
$\tilde{z}$	Short range weighted average of z
i	Subscript for input units
j	Subscript for hidden units
w_{ij}	Weight from input unit i to hidden layer unit j
w_{jk}	Weight from hidden layer unit j to output layer unit k
Δw_{ij}	Weight change from unit i to unit j
θ_j	Bias weight to hidden unit j
a, b, k	Subscript for output units
$\tilde{\Phi}$	Short range kernel for convolution to form $\tilde{z}$
$\bar{\Phi}$	Long range kernel for convolution to form $\bar{z}$
h_u, h_v	Short and long range half lives for convolution kernels

A.2 Calculating Network Activation

Unit activations, x_j, and outputs, z_j, of hidden layer units are calculated by:

$$x_j = \sum_i w_{ij} z_i + \theta_j$$

$$z_j = \tanh(x_j)$$

The output values of hidden units are then used to compute the activation and output value of the one output unit:

$$x_k = \sum_j w_{jk} z_j$$
$$z_k = x_k \qquad \text{(Identity transfer function)}$$

A.3 Calculating the Merit Function

The merit function, F, is defined as:

$$F = \log \frac{V}{U}$$

where V is the long range variance and U is the short range variance:

$$V = \frac{1}{2} \sum_k (\bar{z}_k - z_k)^2 \tag{1}$$
$$U = \frac{1}{2} \sum_k (\tilde{z}_k - z_k)^2 \tag{2}$$

$\bar{z}_k$ is the long range mean, and $\tilde{z}_k$ is the short range mean for an output unit k, defined by a convolution process:

$$\bar{z}_k = \sum_a \bar{\Phi}_{a-k} z_a \tag{3}$$
$$\tilde{z}_k = \sum_a \tilde{\Phi}_{a-k} z_a \tag{4}$$

where a ranges over all virtual output units.

To calculate the derivative of the merit function with respect to each weight, we need to determine:

$$\frac{\partial F}{\partial w} = \frac{1}{V}\frac{\partial V}{\partial w} - \frac{1}{U}\frac{\partial U}{\partial w}$$

Hence, after all of the input has been presented to the network, we calculate U and V. We then need to calculate $\frac{\partial U}{\partial W}$, or to use the back propagation approach (Rumelhart, Hinton, & Williams, 1986), we need to calculate $\frac{\partial U}{\partial x_a}$.

A.4 Deriving the Error Derivatives

In a similar fashion to the Back Propagation algorithm, once the errors δ are known for units in the output layer, they can then be used to calculate weight changes between output and hidden layer units. The errors can also then be propagated back to the hidden layer units so that the weights between input and hidden layer units can be updated.

Given δ_k for output layer units, we can then calculate:

$$\Delta w_{jk} = \delta_k z_j \tag{5}$$

We can also calculate δ_j from the δ_k:

$$\delta_j = g'(x_j) \sum_k w_{jk} \delta_k \tag{6}$$

where $g'(x)$ is the derivative of the activation function for the hidden layer units.

A.4.1 Computing $\frac{\partial U}{\partial x_a}$

The error δ attributed to a unit is defined to be $\delta_a = \frac{\partial E}{\partial x_a}$ in normal back propagation, where E is the overall error of the network, and so we need to find $\frac{\partial F}{\partial x_a}$:

$$\text{given} \quad F = \log \frac{V}{U} \tag{7}$$

$$\frac{\partial F}{\partial x_a} = \frac{1}{V}\frac{\partial V}{\partial x_a} - \frac{1}{U}\frac{\partial U}{\partial x_a} \tag{8}$$

And so to find $\frac{\partial F}{\partial x_a}$, we need to find $\frac{\partial U}{\partial x_a}$ and $\frac{\partial V}{\partial x_a}$:

$$U = \frac{1}{2}\sum_k (\bar{z}_k - z)^2 \tag{9}$$

$$\frac{\partial U}{\partial x_a} = \sum_k (\bar{z}_k - z)\left(\frac{\partial \bar{z}_k}{\partial x_a} - \frac{\partial z_k}{\partial x_a}\right) \tag{10}$$

To find $\frac{\partial z_k}{\partial x_a}$:

$$z_k = f(x_k) \tag{11}$$

Here the transfer function of the output units is the identity function, and so:

$$\frac{\partial z_k}{\partial x_a} = \begin{cases} 1 & \text{if } a = k \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

To find $\frac{\partial \tilde{z}_k}{\partial x_a}$:

$$\tilde{z}_k = \sum_b \tilde{\Phi}_{b-k} z_b \tag{13}$$

$$\frac{\partial \tilde{z}_k}{\partial x_a} = \sum_b \tilde{\Phi}_{b-k} \frac{\partial z_b}{\partial x_a} \tag{14}$$

Using Equation 12, we can then see that equation 14 is zero unless $a = b$:

$$\frac{\partial \tilde{z}_k}{\partial x_a} = \tilde{\Phi}_{a-k} \tag{15}$$

The values for $\frac{\partial z_k}{\partial x_a}$ and $\frac{\partial \tilde{z}_k}{\partial x_a}$ from equations 12 and 15 can then be substituted back into equation 10 to give:

$$\frac{\partial U}{\partial x_a} = \begin{cases} \sum_k (\tilde{z}_k - z)(\tilde{\Phi}_{a-k} - 1) & \text{if } a = k \\ \sum_k (\tilde{z}_k - z)(\tilde{\Phi}_{a-k}) & \text{otherwise} \end{cases} \tag{16}$$

So, if we simply define a new kernel $\tilde{\Phi}'$:

$$\tilde{\Phi}'_a = \begin{cases} \tilde{\Phi}_a - 1 & \text{if } a = 0 \\ \tilde{\Phi}_a & \text{otherwise} \end{cases} \tag{17}$$

$$\frac{\partial U}{\partial x_a} = \sum_k (\tilde{z}_k - z_k)\tilde{\Phi}'_{a-k} \tag{18}$$

By symmetry, a similar expression can be derived for $\frac{\partial V}{\partial x_a}$, so that the final δ_a is:

$$\delta_a = \frac{\partial F}{\partial x_a} = \frac{1}{V}\frac{\partial V}{\partial x_a} - \frac{1}{U}\frac{\partial U}{\partial x_a} \tag{19}$$

A.4.2 Summary: Creating weight changes Δw

1. Calculate the output of virtual output units.
2. Calculate U and V using equations 1 and 2
3. Calculate δ_a for all output units using equations 18 and 19.
4. Back propagate errors to hidden layer units using equation 6.
5. Calculate weight changes $\Delta w_{uv} = \delta_v z_u$ for all weights.

A.5 Convolution Masks

For these experiments, exponential masks were used to create the long and short range weighted averages. The short and long range convolution masks are created by the same equations, but using different values for λ:

$$\begin{aligned}
&\text{2D:} \quad \tilde{\Phi}(x,y) = \exp^{-\lambda_u(x^2+y^2)}, \quad x,y \in [-w,+w] \\
&\text{1D:} \quad \tilde{\Phi}(x) = \exp^{-\lambda_u(x^2)}, \quad x \in [-w,+w]
\end{aligned}$$

λ controls the spread of the function. Here we have defined λ in terms of its half life h:

$$\lambda_u = \frac{\ln 2}{h_u}$$

The size of the masks is defined by the value of w. Here we have set the cut off point for the exponentials to be four half lives from the centre of the mask:

$$w = \begin{cases} 4h & \text{if } 4h < \frac{nx}{2} - 1 \\ \frac{nx}{2} - 1 & \text{otherwise} \end{cases}$$

where nx is the width of the virtual output array. (This upper limit on the mask width is to prevent the mask being bigger than the virtual output array.) The masks are also divisively normalised so that the sum of the mask elements is equal to 1.0.

References

Becker, S. (1992). *An Information-Theoretic Unsupervised Learning Algorithm for Neural Networks*. Ph.D. thesis, Graduate Department of Computer Science, University of Toronto.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Stone, J. V. (1994). Learning spatio-temporal invariances. Tech. rep. CSRP 330, COGS, University of Sussex.

Stone, J. V. (1995). A canonical microfunction for learning perceptual invariances. *Perception, forthcoming. Also available as CSRP 398.*

Stone, J. V., & Bray, A. (1995). A learning rule for extracting spatio-temporal invariances. *Network*, *6*(3), 429–436.

Williams, P. (1991). A marquardt algorithm for choosing the step-size in backpropagation learning with conjugate gradients. Tech. rep. CSRP 229, COGS, University of Sussex.