

Learning transform invariant object recognition in the visual system with multiple stimuli present during training

S.M. Stringer, E.T. Rolls*

Oxford University, Centre for Computational Neuroscience, Department of Experimental Psychology, South Parks Road, Oxford OX1 3UD, England, United Kingdom

ARTICLE INFO

Article history:

Received 19 September 2006

Accepted 7 November 2007

Keywords:

Object recognition

Inferior temporal cortex

Competitive neural networks

Continuous transformation learning

Trace learning

ABSTRACT

Over successive stages, the visual system develops neurons that respond with view, size and position invariance to objects or faces. A number of computational models have been developed to explain how transform-invariant cells could develop in the visual system. However, a major limitation of computer modelling studies to date has been that the visual stimuli are typically presented one at a time to the network during training. In this paper, we investigate how vision models may self-organize when multiple stimuli are presented together within each visual image during training. We show that as the number of independent stimuli grows large enough, standard competitive neural networks can suddenly switch from learning representations of the multi-stimulus input patterns to representing the individual stimuli. Furthermore, the competitive networks can learn transform (e.g. position or view) invariant representations of the individual stimuli if the network is presented with input patterns containing multiple transforming stimuli during training. Finally, we extend these results to a multi-layer hierarchical network model (VisNet) of the ventral visual system. The network is trained on input images containing multiple rotating 3D objects. We show that the network is able to develop view-invariant representations of the individual objects.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

An important problem in understanding natural vision is how the brain can build invariant representations of individual objects even when multiple objects are present in a scene. What mechanisms enable the learning to proceed without the different objects interacting with each other to interfere with the learning of individual object representations? In this paper we describe and analyze an approach to this which relies on the statistics of natural environments. The approach takes account of the statistics that any object can be present with any one of a number of other objects or backgrounds during learning, with some statistical independence of any one object from other objects in the scene.

Over successive stages, the visual system develops neurons that respond with view, size and position (translation) invariance to objects or faces (Desimone, 1991; Perrett & Oram, 1993; Rolls, 1992, 2000; Rolls & Deco, 2002; Tanaka, Saito, Fukada, & Moriya, 1991). For example, it has been shown that the inferior temporal visual cortex has neurons that respond to faces and objects with translation (Ito, Tamura, Fujita, & Tanaka, 1995; Kobatake & Tanaka, 1994; Op de Beeck & Vogels, 2000; Tovee, Rolls, &

Azzopardi, 1994), size (Ito et al., 1995; Rolls & Baylis, 1986), and view (Booth & Rolls, 1998; Hasselmo, Rolls, Baylis, & Nalwa, 1989) invariance. Such invariant representations, once learned by viewing a number of the transforms of the object, are then useful in the visual system for allowing one-trial learning of, for example, the stimulus–reward association of the object, to generalize to other transforms of the same object (Rolls, 2005; Rolls & Deco, 2002).

A number of computational models have been developed to explain how transform-invariant cells could develop in the visual system (Fukushima, 1980; Riesenhuber & Poggio, 1999; Rolls, 2008; Wallis & Rolls, 1997). Two major theories which have sought to explain how transform-invariant representations could arise through unsupervised training with real-world visual input are *trace learning* (Földiák, 1991; Rolls & Milward, 2000; Wallis & Rolls, 1997), and *Continuous Transformation (CT) learning* (Perry, Rolls, & Stringer, 2006; Stringer, Perry, Rolls, & Proske, 2006). Trace learning relies on the temporal continuity of visual objects in the real world (as does slow feature analysis (Wiskott & Sejnowski, 2002)), while in contrast, CT learning relies on spatial continuity.

In most previous studies, however, of invariance learning in hierarchical networks that model the ventral visual stream, only one stimulus is presented at a time during training (Rolls & Milward, 2000; Rolls & Stringer, 2006; Stringer et al., 2006; Wallis & Rolls, 1997). In this paper we investigate whether, and if so how, models of this type can self-organize during training when multiple stimuli are presented together within each visual image.

* Corresponding author. Tel.: +44 1865 271419; fax: +44 1865 310447.

E-mail addresses: simon.stringer@psy.ox.ac.uk (S.M. Stringer), Edmund.Rolls@oxcns.org (E.T. Rolls).

URLs: <http://www.oftnai.org> (S.M. Stringer), <http://www.oxcns.org> (E.T. Rolls).

In Section 3 we show how a standard 1-layer competitive network responds when trained on input patterns containing multiple, e.g. pairs of, independent stimuli. As the number of stimuli N increases, the number of possible input patterns which are composed of pairs of stimuli $N(N - 1)/2$ grows quadratically. For small numbers of stimuli N , the output neurons still represent the paired-stimulus input patterns. However, for large enough N , the output neurons begin to learn to respond to the individual stimuli instead of the multi-stimulus input patterns used during training. In this way, we show that a standard competitive network may suddenly switch from learning representations of the multi-stimulus input patterns to representing the individual stimuli as the number of independent stimuli grows large enough.

In Section 4 we continue to investigate how a 1-layer competitive network may learn to process input patterns containing multiple stimuli. However, we now extend the simulations by allowing the independent stimuli to transform, e.g. translate across the input space, during training. We show that, even when the network is trained on input patterns containing pairs of transforming stimuli, a standard 1-layer competitive network is able to learn invariant representations of the individual stimuli.

In Section 5 we extend these results to a full 4-layer hierarchical feedforward network model (VisNet) of the ventral visual processing stream. The visual input stimuli are rotating 3D objects created using the OpenGL 3D image generation software. We train the network on input images containing multiple rotating objects. We demonstrate that the network is able to develop view-invariant representations of the individual objects.

2. A hypothesis on how learning can occur about single objects even when multiple objects are present

We consider how learning about individual objects can occur even when a number of objects are present. Consider a situation that might occur in the real world in which an individual object is present, but is accompanied by one other object from a set of other objects. A possible series of trials might have (where each number refers to an individual object) all possible pairs of objects. For the case of $N = 4$ stimuli, there are a total of 6 paired-stimulus training patterns used on different trials as follows: 1 + 2, 1 + 3, 1 + 4, 2 + 3, 2 + 4 and 3 + 4. We approach the categorization that would occur using a competitive network, as this type of network performs categorization, is biologically plausible, and is the prototypical network involved in hierarchical models of object recognition (Hertz, Krogh, & Palmer, 1991; Rolls & Deco, 2002). We are interested in whether the network forms representations of the object pairs actually shown during training, or whether it can form, as desired for this application, separate representations of each individual object.

Consider the associations between the activity of the input neurons for each stimulus. For stimulus 1, the set of neurons that provide the representation of stimulus 1 have high correlations with each other. Moreover, these are higher correlations than those between the neurons representing different stimuli. (This occurs because on different trials the neurons representing stimulus 1 are sometimes paired with those representing stimulus 2, sometimes with those representing stimulus 3, etc.) We can calculate the number of times that the neurons within a stimulus are active simultaneously during one training epoch in which all pairs are presented, and with N stimuli, this is $N - 1$. In contrast, the neurons from different stimuli are co-active on only one occasion in a training epoch. The ratio of the within-stimulus to between-stimulus co-occurrences is thus $N - 1 : 1$. This increases with the number of stimuli. Given that competitive networks effectively build categories based on correlations between sets of afferents to the network, we propose that as N increases, a stage will

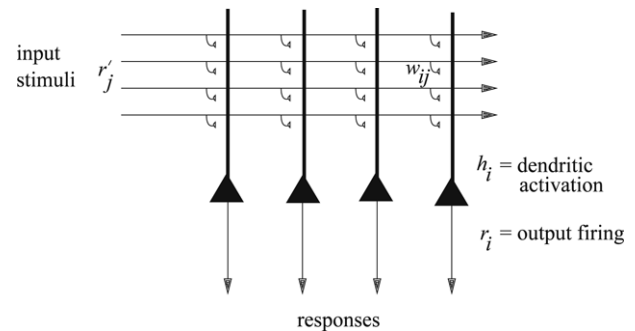


Fig. 1. The architecture of a competitive neural network. There is an input layer of cells with feedforward associatively modifiable synaptic connections onto an output layer of cells. At each timestep during learning, an input pattern is applied to the layer of input cells. Next, activity from the input layer is propagated through the feedforward connections to activate a winning set of cells in the output layer. Within the output layer there is feedback inhibition implemented by inhibitory neurons not shown in the figure, which ensures that only a small subset of output cells remains active. Next, the synaptic weights between the active input cells and the active output cells are strengthened. In this way, the output cells become associated with particular patterns of activity in the input layer.

be reached where instead of representing the pairs of stimuli actually presented during training, the network will instead form representations of the individual stimuli. We examine whether this transition occurs in Section 3.

3. Training a 1-layer competitive network with input patterns which contain multiple independent stimuli

In this section we show how the mechanism described in Section 2 operates in a 1-layer competitive network.

3.1. Model

The neural network architecture is shown in Fig. 1. The fully connected 1-layer model has the architecture of a standard competitive network (Hertz et al., 1991; Rolls & Deco, 2002). There is an input layer of cells with feedforward associatively modifiable synaptic connections onto an output layer of cells. At each timestep during learning, an input pattern is applied to the layer of input cells. Next, activity from the input layer is propagated through the feedforward connections to activate a set of cells in the output layer. Within the output layer there is competition implemented by feedback inhibition and the threshold nonlinearity of neurons. Next, the synaptic weights between the active input cells and the active output cells are strengthened by associative (Hebbian) learning. The output cells self-organize to represent and thus categorize different patterns of activity in the input layer.

The competitive network contained 100 input cells, 100 output cells, and was fully connected. Each simulation experiment used a set number N of independent stimuli. The stimulus patterns are represented by the activation of non-overlapping contiguous blocks of $100/N$ cells. Stimulus 1 is represented by cells 1 to $100/N$, stimulus 2 by the next block of $100/N$ cells, and so on up to N stimuli. For the case $N = 4$, Fig. 2 (left) shows the input representations of the 4 independent stimuli. (In this paper, by ‘independent stimuli’ we refer to stimuli of the type labelled as 1–4 in Fig. 2 left, and note that each such stimulus is paired with each of the other stimuli equally often during the training.) Fig. 2 (right) shows the 6 paired-stimulus input patterns that were used to train the 1-layer competitive network for the case $N = 4$, in which the training patterns are: 1 + 2, 1 + 3, 1 + 4, 2 + 3, 2 + 4 and 3 + 4.

During training, the agent is presented in turn with each of the possible $N(N - 1)/2$ paired-stimulus input patterns. At the presentation of each training pattern, the input cells are set to fire

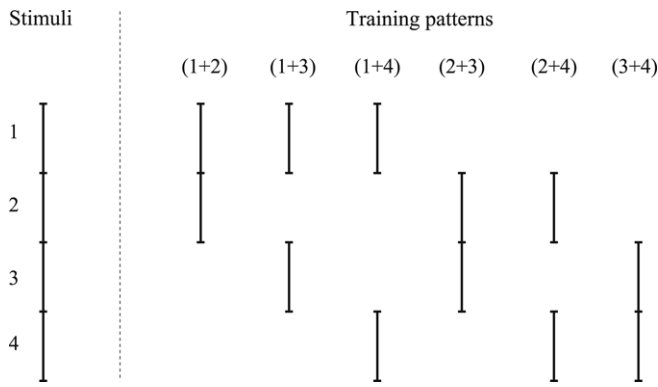


Fig. 2. The input representations of the independent stimuli and the paired-stimulus input patterns used to respectively test and train the 1-layer competitive network. The competitive network has 100 input cells arranged in a vertical line. Each simulation experiment uses a set number of independent stimuli N . The stimulus patterns are represented by the activation of non-overlapping contiguous blocks of $100/N$ cells. Stimulus 1 is represented by cells 1 to $100/N$, stimulus 2 by the next block of $100/N$ cells, and so on up to N stimuli. The case of $N = 4$ stimuli is represented in the figure. Also shown are the corresponding 6 paired-stimulus training patterns. For the case of $N = 4$ stimuli, there are a total of 6 paired-stimulus training patterns as follows: 1 + 2, 1 + 3, 1 + 4, 2 + 3, 2 + 4 and 3 + 4. The cells that are active in each pattern are shown as black.

according to the paired-stimulus activity training patterns shown in Fig. 2 (right). The activity from the input layer is then propagated through the feedforward synaptic connections to activate a set of cells in the output layer. (The synaptic weights from the input to output cells are initially set to random values from a uniform distribution in the range 0–1, and then normalized to a vector length of 1. This is standard in competitive networks, and ensures that some firing of output cells will be produced by the inputs, with each output cell likely to fire at a different rate for any one input stimulus.) The activations of the cells in the output layer are calculated according to

$$h_i = \sum_j w_{ij} r_j, \quad (1)$$

where h_i is the activation of output cell i , r_j is the firing rate of input cell j , and w_{ij} is the synaptic weight from input cell j to output cell i .

The activation h_i of each neuron was converted to the firing rate r_i of each neuron using a threshold linear activation, and adjusting the threshold to achieve a prescribed sparseness of the representation in the output cells. (This represents a process by which mutual inhibition between the output cells through inhibitory interneurons implements competition to ensure that there is only a small winning set of output cells left active.) The sparseness a of the representation that was prescribed was defined, by extending the binary notion of the proportion of neurons that are firing, as follows

$$a = \frac{\left(\sum_{i=1}^M r_i / M \right)^2}{\sum_{i=1}^M r_i^2 / M}, \quad (2)$$

where r_i is the firing rate of the i th neuron in the set of M neurons (Rolls & Treves, 1990, 1998). In the simulations, the competition was achieved in an iterative cycle by feedback adjustment of the threshold for the firing of neurons until the desired sparseness was reached. The threshold was the same for all neurons. With the graded firing rates of the neurons, the result was that typically the proportion of neurons with non-zero firing rates after the competition was numerically somewhat larger than the sparseness value given as a parameter to the network.

Next, the synaptic weights between the active input cells and the active output cells are strengthened according to the associative Hebb learning rule

$$\delta w_{ij} = k r_i r_j \quad (3)$$

where δw_{ij} is the change of synaptic weight, and k is the learning rate constant which was set to 0.001 unless otherwise stated. To prevent the same few neurons from always winning the competition, the synaptic weight vectors are set to unit length after each learning update for each training pattern. To implement weight normalization the synaptic weights were rescaled to ensure that for each output cell i we have

$$\sqrt{\sum_j (w_{ij})^2} = 1, \quad (4)$$

where the sum is over all input cells j . Such a renormalization process may be achieved in biological systems through heterosynaptic long-term depression that depends on the existing value of the synaptic weight (Oja, 1982; Rolls & Deco, 2002; Rolls & Treves, 1998). (Heterosynaptic long-term depression in the brain was described by Levy (1985) and Levy and Desmond (1985); see Brown, Kairiss, and Keenan (1990)).

The presentation of all possible $N(N - 1)/2$ stimulus-pair training patterns corresponded to one training epoch. For each experiment with fixed N , there were 10,000 training epochs to ensure convergence of the synaptic weights.

For each experiment, after training, the network was tested with N single-stimulus input patterns, each of which contained a different one of the independent stimuli as illustrated at the left of Fig. 2. During this testing, for each output cell we recorded how many of the N stimuli the cell responded to. The output neurons tended to be binary, that is to either be firing with a high rate or not at all for a given stimulus (as will be evident when Figs. 3 and 4 are described). An output cell was classed as responsive to a particular input stimulus if presentation of the input stimulus elicited a firing rate in the output cell which was greater than 50% of the maximal firing rate of any output cell to any stimulus.

3.2. Simulation results

The results described next showed that if $N \leq 5$ the network represented every one of the $N(N - 1)/2$ paired-stimulus input patterns, and generally none of the N single-stimulus input patterns. If $N \geq 6$ the network represented generally none of the $N(N - 1)/2$ paired-stimulus input patterns, and all of the N single-stimulus input patterns. These results are shown in Table 1. We remind the reader that the network was in all cases trained on the $N(N - 1)/2$ paired-stimulus input patterns.

Table 1 summarizes the results obtained from simulation experiments, which show how the output cells of the 1-layer competitive network learn to represent the input data when different numbers of independent stimuli N are used to generate multi-stimulus training patterns. For the experimental results shown in Table 1, the population sparseness of the output firing rates a was always set to 0.05, and the average results obtained over 6 separate simulation experiments for a particular fixed value of N are shown. In all of the simulation experiments, after training, the output cells were found to be responsive to either 0 stimuli, a single stimulus, or 2 stimuli. No cells were ever found which responded to 3 or more stimuli. The standard errors of the means shown in Table 1 were in most cases less than 1. In all these simulations approximately equal numbers of cells learned to respond to each of the single stimuli for $N \geq 6$ or paired stimuli if $N \leq 5$ (as will be made evident in Fig. 4).

The ability of the network to learn to represent the independent component stimuli when N is large is quite robust with respect

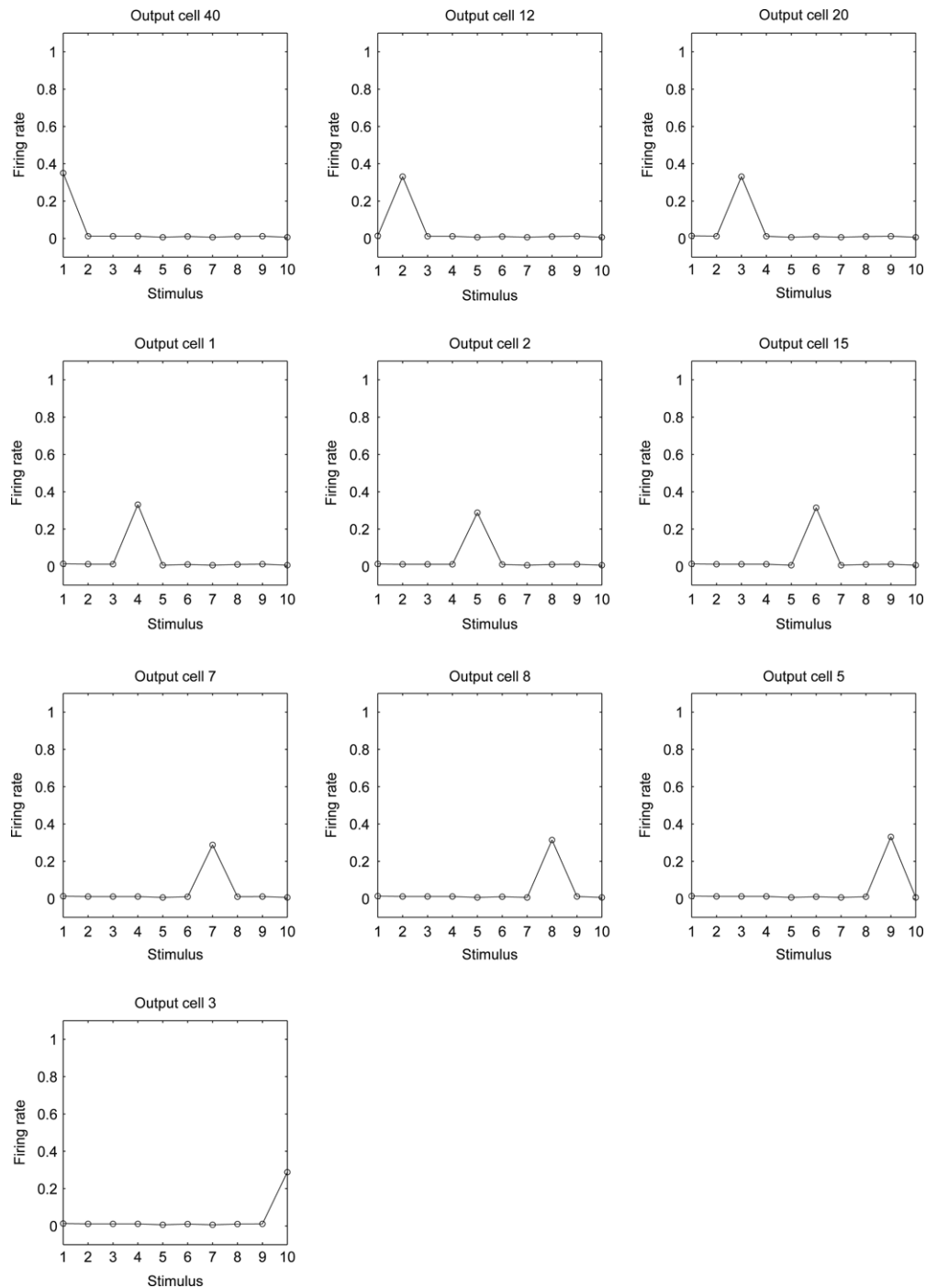


Fig. 3. The firing rate responses of a selection of 10 output cells to $N = 10$ independent stimuli, after the network has been trained on all possible stimulus-pair patterns constructed from the stimuli. The sparseness of the output firing rates was set to 0.2 during training and testing. For this test case, all 100 output cells learned to respond to a single stimulus, with disjoint subsets of approximately 10 output cells having learned to respond to each of the independent stimuli. This figure shows the responses of a selection of 10 output cells, each of which had learned to respond to a different one of the 10 stimuli.

to the sparseness of the output firing rates a , as shown by the results obtained for different values of the sparseness and $N = 10$ in Table 2. For the wide range of sparseness values tested, there were always mainly cells that responded only to a single independent stimulus, and rarely more than a few cells that learned to respond to the paired-stimulus training patterns. As before, the average values for 6 runs are shown, and the standard errors were small. With small values of the sparseness parameter a , only a few neurons were activated by a given stimulus, and most of the neurons therefore remained unallocated to any stimulus, retaining their initial random synaptic weights. With $a = 0.2$

approximately 20 of the 100 output cells were activated by each input stimulus-pair pattern during training. This corresponds to each single stimulus activating approximately 10 cells, and this is why as shown in Table 2 all 100 output cells became responsive to only a single stimulus after training, with a different set of 10 output cells associated with each of the 10 stimuli. Moreover, the number of output neurons is not the factor that determines the representations formed, for when the transition occurs to represent individual stimuli (see Table 1), the number of stimuli N equals 6, the number of stimulus pairs is 15, and the number of output neurons is 100.

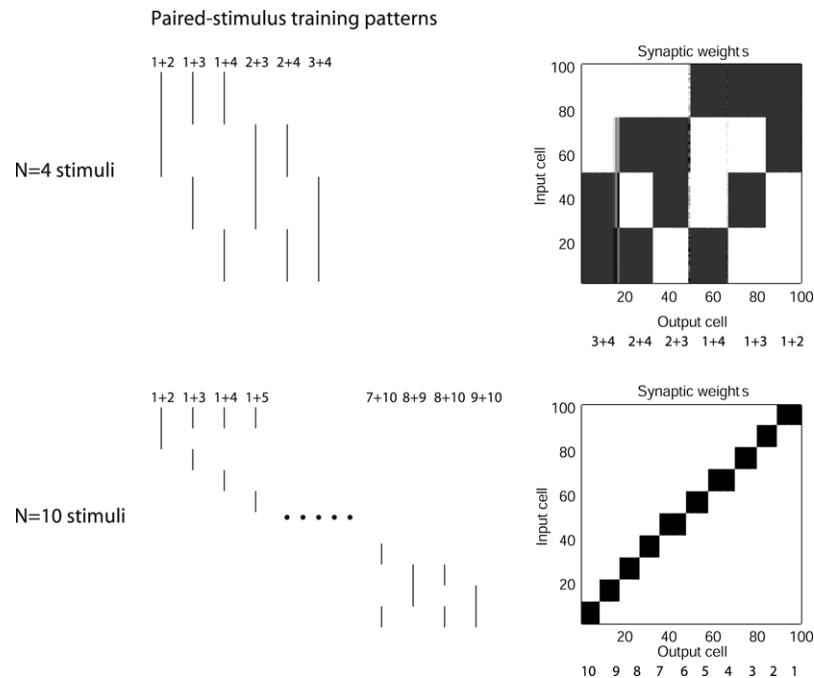


Fig. 4. The synaptic weights after the network has been trained with paired-stimulus input patterns constructed from different numbers of stimuli N . The top row shows results for the case $N = 4$ stimuli with $a = 0.14$. On the left is shown the 6 paired-stimulus input patterns presented during training (using the same conventions as in Fig. 2), while on the right is shown the synaptic weight matrix within the network after training. In the weight matrix plots, dark shading indicates a high weight value. In the plot of the synaptic weights, the output cells have been ordered to reveal the underlying weight structure which developed during training. The synaptic weights show that the output cells have typically learned to respond to pairs of stimuli, which correspond to the 6 original paired-stimulus input patterns presented during training. The bottom row shows results for the case $N = 10$ stimuli with $a = 0.2$. On the left are shown some of the 45 paired-stimulus training patterns, while on the right is shown the synaptic weight matrix after training. As above, the output cells have been ordered to reveal the underlying weight structure which developed during training. It can be seen that the weight matrix has a block-diagonal structure, which clearly shows that the output cells have each learned to respond to one particular stimulus.

Table 1

A summary of how the output cells of the 1-layer competitive network learn to represent the input data when different numbers of independent stimuli N are used to generate multi-stimulus training patterns

Number of stimuli N	Number of cells which respond to 1 stimulus	Number of cells which respond to 2 stimuli
3	0.2	18.0
4	0.0	36.0
5	0.0	60.0
6	12.3	0.0
7	18.8	0.0
8	27.5	0.2
9	35.7	0.3
10	44.8	1.5

For each value of N , average results were obtained over 6 simulation runs. For these experiments, the sparseness of the output firing rates a was always set to 0.05. It can be seen that when there are a small number of stimuli, i.e. for $N = 3$ to 5, the output cells typically learn to respond to pairs of stimuli, corresponding to the multi-stimulus input patterns which the network was exposed to during training. However, when the number of stimuli N is increased to 6 or more, the output cells typically learn to respond to only a single one of the stimuli. Thus, for large numbers of stimuli N , the output cells of the competitive network learn to represent the N individual independent stimuli rather than the $N(N-1)/2$ multi-stimulus training patterns, which the network was exposed to during training.

Examples of the firing rates of the output neurons in these simulations after training are shown in Fig. 3 (obtained for the case, $N = 10$ stimuli and the output firing rate sparseness set to 0.2). In each plot, the cell responds to only one of the 10 stimuli.

The behavior of the competitive network can be further understood from the synaptic weight matrices shown in Fig. 4. The top row shows results for the case, $N = 4$ stimuli with $a = 0.14$. The synaptic weights show that the output cells have typically learned to respond to pairs of stimuli, which correspond to the 6 original paired-stimulus input patterns presented during training. The bottom row shows results for the case, $N = 10$ stimuli with $a = 0.2$. It can be seen that the weight matrix has a block-diagonal

Table 2

A summary of how the output cells of the 1-layer competitive network learn to represent the input data when the level of mutual inhibition within the output layer is varied in order to control the sparseness of the output firing rates a

Sparseness of output firing a	Number of cells which respond to 1 stimulus	Number of cells which respond to 2 stimuli
0.01	11.2	0.0
0.02	14.7	0.0
0.05	45.5	1.0
0.1	69.3	9.3
0.2	100.0	0.0
0.5	66.3	0.3

For these experiments, the number of independent stimuli N was always set to 10. It can be seen that, with a relatively large number of $N = 10$ stimuli, the output cells mostly learned to respond to only a single one of the stimuli across a wide range of values of the sparseness of the output firing rates a . However, the number of output cells which responded to a single stimulus was maximized when the output sparseness a was set to 0.2, where it can be seen that all 100 output cells are responsive to only a single stimulus.

structure, which clearly shows that the output cells have each learned to respond to only one of the independent stimuli. Since the blocks on the diagonal are almost square, the weight matrix also confirms that the output cells are distributed fairly evenly among the 10 stimuli, with disjoint subsets of approximately 10 output cells having learned to respond to each of the 10 independent stimuli.

The simulations described above were performed with 10,000 training epochs to ensure convergence during learning. However, similar effects can still be observed after far fewer training epochs. We repeated an experiment with $N = 10$ independent stimuli and the sparseness of the output firing rates set to 0.2. However, this time we increased the learning rate to $k = 0.01$ and used only 100 training epochs. This led to identical results to those shown in row 5 of Table 2. Thus, 100 training epochs were enough to ensure that all of the output cells learned to respond to a single

