Supervised Architectures for Internal Simulation of Perceptions and Actions

Magnus Johnsson, David Gil, Christian Balkenius and Germund Hesslow

Abstract We present a study of supervised neural network architectures capable of internal simulation of perceptions and actions. These architectures employ the novel Associative Self-Organizing Map (A-SOM) as a hidden layer (for the representation of perceptions), and a neural network adapted by the delta rule as an output layer (for the representation of actions). The A-SOM develops a representation of its input space, but in addition it also learns to associate its activity with an arbitrary number of additional (possibly delayed) inputs. We test architectures, with as well as without, recurrent connections. The simulation results are very encouraging. The architecture without recurrent connections correctly classified 100% of the training samples and 80% of the test samples. After ceasing to receive any input the best of the architectures with recurrent connections was able to continue to produce 100% correct output sequences for 28 epochs (280 iterations), and then to continue with 90% correct output sequences until epoch 42.

1 Introduction

It has been suggested that humans and maybe other higher animals are able to internally simulate interactions with their environment. According to the simulation hypothesis [6] this can be done by utilizing three proposed mechanisms. First (sim-

Magnus Johnsson
Lund University Cognitive Science, Sweden, e-mail: Magnus.Johnsson@lucs.lu.se

David Gil
Computing Technology and Data Processing, University of Alicante, Spain e-mail: dgil@dtic.ua.es

Christian Balkenius
Lund University Cognitive Science, Sweden, e-mail: Christian.Balkenius@lucs.lu.se

Germund Hesslow
Department of Experimental Medical Science, Lund, Sweden, e-mail: Germund.Hesslow@med.lu.se

ulation of actions), it is assumed that brain activity can occur that resembles the activity that normally occurs when actions are performed, except that the motor output is suppressed during simulation. Second (perceptual simulation), the brain can elicit activity in sensory cortex that resembles the activity that would normally occur as a consequence of sensory input. The third assumption (anticipation) is that both overt and merely simulated actions can elicit perceptual simulation of the probable consequences of the action. (Simulated perception may also be elicited by sensory activity in a different modality.) In overt interaction with the environment, the consequences of an action generate sensory input that can function as stimuli for new actions. In simulated interaction (Fig 1 b), simulated actions elicit sensory consequences via associatively learned perceptual simulations rather than via the physical consequences and the sense organs (Fig 1 a). An internal simulation might start with a real perception and continue with a chain of simulated actions and perceptions. Obviously this ability has survival value, since it provides the animal with a way of evaluating a potential course of action before acting it out in the physical world with perhaps lethal consequences.

An artificial agent acting on its own in the real world should have this ability as well. A good question is how to implement it. One way to do it would be a connectionist approach, i.e. to try to find a suitable neural network architecture with this ability. Ziemke et al [12] have done such experiments together with a simulated Khepera robot, using a genetic algorithm to set the values of the weights.

Previously we have focused on the perceptual side of the problem. Thus we have done experiments with a novel self-organizing neural network called the Associative Self-Organizing Map (A-SOM). The A-SOM is an extension of the SOM [11], which learns to associate its activity with additional inputs. This was done in simulations with an A-SOM which received the activities of two external SOMs as additional inputs [9], and in the context of haptic perception were we implemented a bio-inspired self-organizing texture and hardness perception system which automatically learned to associate the self-organized representations of these two submodalities (A-SOMs) with each other [7] [8]. We have also done experiments with a system consisting of two connected A-SOMs [10]. One of these A-SOMs also

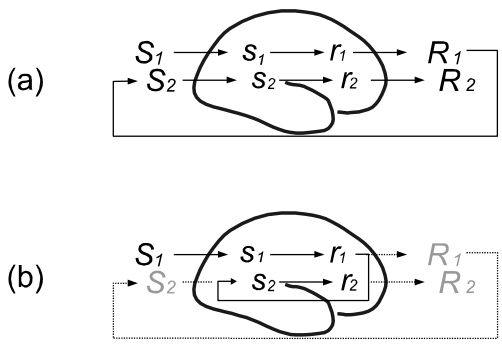

Fig. 1 a. Real interaction with environment. Stimulus S_1 causes perceptual activity s_1, which causes preparatory response r_1 and overt response R_1. R_1 causes predictable new stimulus S_2, which causes new sensory activity etc. b. Simulated interaction. Preparatory response r_1 elicits, via internal association mechanisms, perceptual activity s_2 before overt behaviour occurs and causes new stimulus.

learned to associate its current activity with its activity in the previous iteration. This created a novel kind of recurrent Self-Organizing Map which was able to learn perceptual sequences. This activity sequence could be invoked by input to either of the two A-SOMs or both.

In this paper we take the next step and add an action layer to the A-SOM, thus obtaining an architecture able to simulate chains of both perceptions and actions. This implies an architecture which can elicit reasonable sequences of activity both in its perceptual network (hidden layer) and in its action network (output layer). We present four supervised neural network architectures that employ the A-SOM as a hidden layer together with an output layer adapted by the delta rule. One of these architectures only has feed-forward connections and is suitable for classification tasks. The other three architectures in addition use recurrent connections that make them able to continue with internal simulations of both perceptions and actions in the absence of input. These architectures are related to the well known recurrent supervised neural network architectures of Jordan and Elman [3] but adds the properties of the SOM.

The implementation of all code for the experiments presented in this paper was done in C++ using the neural modeling framework Ikaros [1].

2 Neural Network Architectures

All architectures discussed in this paper have a common basic structure. Thus they consist of two layers (actually two separate but connected neural networks), i.e. a hidden layer and an output layer. The hidden layer consists of an A-SOM which is fully connected with forward connections to the output layer (with a time delay of one iteration during a simulation). The output layer consists of a grid of neurons that are adapted by the delta rule to get an activity that converges to the provided desired output. The different architectures discussed differ in whether they include recurrent feedback connections and if so how these are connected.

2.1 The Hidden Layer

The hidden layer consists of an Associative Self-Organizing Map (A-SOM) [9], which can be considered a SOM that learns to associate its activity with (possibly delayed) additional inputs. The A-SOM consists of an $I \times J$ grid of a fixed number of neurons and a fixed topology. Each neuron n_{ij} is associated with $r+1$ weight vectors $w^a_{ij} \in R^n$ and $w^1_{ij} \in R^{m_1}, w^2_{ij} \in R^{m_2}, \ldots, w^r_{ij} \in R^{m_r}$. All the elements of all the weight vectors are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which all the weight vectors are normalized, i.e. turned into unit vectors.

At time t each neuron n_{ij} receives $r+1$ input vectors $x^a(t) \in R^n$ and $x^1(t-d_1) \in R^{m_1}, x^2(t-d_2) \in R^{m_2}, \ldots, x^r(t-d_r) \in R^{m_r}$ where d_p is the time delay for input vector $x^p, p = 1,2,\ldots,r$.

The main net input s_{ij} is calculated using the standard cosine metric

$$s_{ij}(t) = \frac{x^a(t) \cdot w_{ij}^a(t)}{||x^a(t)||||w_{ij}^a(t)||}, \tag{1}$$

The activity in the neuron n_{ij} is given by

$$y_{ij}(t) = \left[y_{ij}^a(t) + y_{ij}^1(t) + y_{ij}^2(t) + \ldots + y_{ij}^r(t)\right] / (r+1) \tag{2}$$

where the main activity y_{ij}^a is calculated by using the softmax function [2]

$$y_{ij}^a(t) = \frac{(s_{ij}(t))^m}{\max_{uv}(s_{uv}(t))^m} \tag{3}$$

where u and v ranges over the rows and the columns of the neural network and m is the softmax exponent.
The ancillary activity $y_{ij}^p(t), \qquad p = 1,2,\ldots,r$ is calculated by again using the standard cosine metric

$$y_{ij}^p(t) = \frac{x^p(t-d_p) \cdot w_{ij}^p(t)}{||x^p(t-d_p)||||w_{ij}^p(t)||}. \tag{4}$$

The neuron c associated with the weight vector $w_c^a(t)$ most similar to the input vector $x^a(t)$, i.e. the neuron with the strongest main activation, is selected:

$$c = \arg\max_c \{|x^a(t) \cdot w_c^a(t)|\} \tag{5}$$

The weights w_{ijk}^a are adapted by

$$w_{ijk}^a(t+1) = w_{ijk}^a(t) + \alpha(t) G_{ijc}(t) \left[x_k^a(t) - w_{ijk}^a(t)\right] \tag{6}$$

where $0 \leq \alpha(t) \leq 1$ is the adaptation strength with $\alpha(t) \to 0$ when $t \to \infty$. The neighbourhood function $G_{ijc}(t) = e^{-\frac{||r_c - r_{ij}||}{2\sigma^2(t)}}$, where $r_c \in R^2$ and $r_{ij} \in R^2$ are location vectors of neurons c and n_{ij}, is a Gaussian function decreasing with time.

The weights $w_{ijl}^p, p = 1,2,\ldots,r$, are adapted by

$$w_{ijl}^p(t+1) = w_{ijl}^p(t) + \beta x_l^p(t-d_p) \left[y_{ij}^a(t) - y_{ij}^p(t)\right] \tag{7}$$

where β is the constant adaptation strength.
All weights $w_{ijk}^a(t)$ and $w_{ijl}^p(t)$ are normalized after each adaptation.

2.2 The Output Layer

The output layer consists of an $I \times J$ grid of a fixed number of neurons and a fixed topology. Each neuron n_{ij} is associated with a weight vector $w_{ij} \in R^n$. All the elements of the weight vector are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which the weight vector is normalized, i.e. turned into unit vectors.

At time t each neuron n_{ij} receives an input vector $x(t) \in R^n$.

The activity y_{ij} in the neuron n_{ij} is calculated using the standard cosine metric

$$y_{ij}(t) = \frac{x(t) \cdot w_{ij}(t)}{||x(t)||||w_{ij}(t)||}, \tag{8}$$

During the learning phase the weights w_{ijl}, are adapted by

$$w_{ijl}(t+1) = w_{ijl}(t) + \beta x_l(t)\,[y_{ij}(t) - d_{ij}(t)] \tag{9}$$

where β is the constant adaptation strength and $d_{ij}(t)$ is the desired activity for the neuron n_{ij}.

2.3 Tested Variants

We have tested four different architectures that use an A-SOM as a hidden layer together with the output layer described above. In the first architecture (Fig 2 a) we fully connected the A-SOM to the output layer with feed-forward connections only, i.e. the output layer received the activity of the A-SOM as input, thus creating an architecture suitable for classification.

The second architecture (Fig 2 b) is similar to the first one but with recurrent connections added. These recurrent connections feed the activity of the A-SOM back to the A-SOM itself as ancillary input with a time delay of one iteration. This yields an architecture able to produce proper sequences of output activity even when the A-SOM stops receiving input.

The third architecture (Fig 2 c) is similar to the second but with the recurrent connections in the second architecture replaced by recurrent connections that feed the activity of the output layer back to the A-SOM as ancillary input with a time delay of one iteration. As the second architecture this also yields an architecture able to produce proper sequences of output activity even when the A-SOM stops receiving input.

In the fourth architecture (Fig 2 d) the approaches of the second and the third architectures are combined, i.e. there are two sets of recurrent connections. Thus there are both a set of recurrent connections that feed the activity of the A-SOM back to the A-SOM itself as ancillary input with a time delay of one iteration, and

a set of recurrent connections that feed the activity of the output layer back to the A-SOM as ancillary input with a time delay of one iteration. Also this architecture are suitable for the production of proper sequences of output activity even when the A-SOM stops receiving input.

3 Simulations

All four tested architectures used an A-SOM with 15×15 neurons as a hidden layer, and an output layer consisting of 1×10 neurons.

To test the architectures we constructed a set of 10 training samples by random selection, with uniform distribution, from a subset s of the plane $s = \{(x,y) \in R^2; 0 \leq x \leq 1, 0 \leq y \leq 1\}$. The selected points were then mapped to a subset of R^3 by adding a third constant element of 0.5, yielding a training set of three-dimensional vectors. The reason for this was that a Voronoi tessellation of the plane was calculated from the generated points to later aid in the determination of how new points in the plane should be classified (the first architecture described above). To make this Voronoi

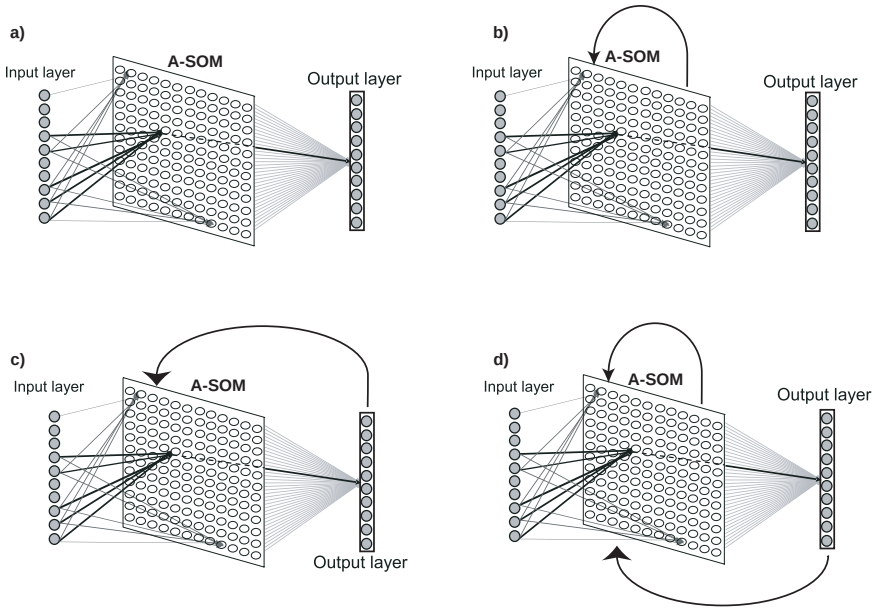

Fig. 2 The four tested neural network architectures. a) The A-SOM is connected to the output layer with feed-forward connections only; b) The A-SOM is connected to the output layer with feed-forward connections and to itself with recurrent connections; c) The A-SOM is connected to the output layer with feed-forward connections, and the output layer is connected with recurrent connections to the A-SOM; d) The A-SOM is connected to the output layer with feed-forward connections and to itself with recurrent connections. In addition the output layer is connected with recurrent connections to the A-SOM.

tessellation, which is based on the Euclidian metric, useful for this purpose with the A-SOM in the hidden layer, which uses a metric based on dot product, the set of points in the plane has to be mapped so that the corresponding position vectors after normalization are unique. One way to accomplish such a mapping is by adding a constant element to each vector. The result of this is that each vector will have a unique angle in R^3. We chose the value 0.5 for the constant elements to maximize the variance of the angles in R^3.

All architectures were trained during 20000 iterations, i.e. during 2000 epochs when receiving the sequence of 10 training samples. The softmax exponent for the A-SOMs were set to 1000. The learning rate $\alpha(0)$ of the A-SOMs was initialized to 0.1 with a learning rate decay of 0.9999 (i.e. multiplication of the learning rate with 0.9999 in each iteration), which means the minimum learning rate, set to 0.01, will be reached at the end of the 20000 training iterations. The neighbourhood radius, i.e. σ of the neighbourhood function $G_{ijc}(t)$ in eq. (6), was initialized to 15 for both A-SOMs and shrunk to 1 during the 20000 training iterations by using a neighbourhood decay of 0.9998 (i.e. multiplication of the neighbourhood radius with 0.9998 in each iteration). The A-SOMs used plane topology when calculating the neighbourhood. The learning rate β for the associative weights in all A-SOMs as well as for the neurons in the output layer for all architectures was set to 0.35.

Since these architectures are supervised each training sample was associated with a desired activity provided to the output layer during the training phase. The set of desired activities D consisted of 10-dimensional vectors, where one element in each was set to 1 and the other elements were set to 0, i.e. $D = \{(1,0,\ldots,0),(0,1,\ldots,0),\ldots,(0,0,\ldots,1)\}$. This means that after training each sample of the training set should elicit the highest activation in a unique neuron if the trained architecture is able to distinguish between the samples in the training set. Moreover, if the trained architecture receives a new sample (this was tested in the case of the first architecture described above), not included in the training set, this should elicit activity in the same output neuron as the closest sample in the training set. In other words: a new sample located in a certain Voronoi cell of the input space should elicit the highest activity in the same neuron as the training sample corresponding to that particular Voronoi cell. If this is true for a sufficient ratio of the new samples, then the generalization ability of the architecture should be considered good.

After the training phase the first architecture described above was tested with the training samples and with an additional set of 10 new samples generated in the same way as the training set.

The other three architectures were tested to evaluate their ability to produce proper sequences of activities in their output layers even when the A-SOMs stopped receiving input. Thus we only used the training sets in this evaluation, and the sequences of activities were considered proper if the same sequence of neurons in the output layer had the highest activity as when the architectures received input. Thus these architectures were tested, after the training phase, by first feeding them the sequence of the 10 training samples once. Then the architectures did not receive any more input and we recorded the activity for the following 950 iterations (see Fig 3

B, C and D) to see if the architectures were able to reproduce the same sequence of 10 output activities over and over again. Thus we recorded the percentage of the iterations in each epoch (i.e. an epoch was 10 iterations since the training sequence was the 10 samples in the training set) with proper activity after the architecture ceased to receive any input.

3.1 Feed-Forward Architecture for Classification

Fig 2 A shows a schematic depiction of the architecture. The training set was used for all four architectures. The simulation results with the first architecture are depicted in Fig 3 A. As can be seen in this figure, all samples in the training set elicited the highest activity in the proper neuron, i.e. 100% correct.

In Fig 3 A we can also see that 8 of the 10 new samples elicited highest activity in the proper neuron of the output layer. Sample 3 in the new sample set should have been classified as belonging to the Voronoi cell for sample 3 of the training set, but was misclassified as belonging to the Voronoi cell for sample 1 in the training set. Sample 7 in the new sample set should have been classified as belonging to the Voronoi cell for sample 7 of the training set, but was misclassified as belonging to the Voronoi cell for sample 4 in the training set.

It is worth noting that sample 10 in the new sample set lies at the border between the Voronoi cells for training samples 4 and 7. This sample was classified as belonging to the Voronoi cell for training sample 7, but it would also be considered correctly classified if it would have been classified as belonging to the Voronoi cell for training sample 4. An interesting observation was that when receiving this new sample the activity of the neuron in the output layer that represents the Voronoi cell for training sample 7 was the most activated neuron in the output layer, and the neuron that represents the Voronoi cell for training sample 4 was the second most activated neuron in the output layer.

3.2 Architecture with Recurrent A-SOM Connections

The simulation results with the second architecture are depicted in Fig 3 B. In this figure we can see that this architecture was able to reproduce the sequence of the training samples with 100% correctness in the first 28 epochs (i.e. for 280 iterations), with 90% correctness until epoch 42, and then there was a gradual decline until it reached a level of 20% correct activities at epoch 76. This still was the performance level at epoch 95.

3.3 Architecture with Recurrent Connections from the Output Layer to the A-SOM

The simulation results with the third architecture are depicted in Fig 3 C. In this figure we can see that this architecture was able to reproduce the sequence of the training samples with 100% correctness in the first 3 epochs (i.e. for 30 iterations), then there was a rapid decline until a level of 0% correct activities were reached at epoch 14. This level of 0% correct activities continued until epoch 71 when it started to improve again. It reached a new peak of 50% correct activities between epochs 79 and 86. After that there was a decline again and the level of 0% correct activities was reached again at epoch 95.

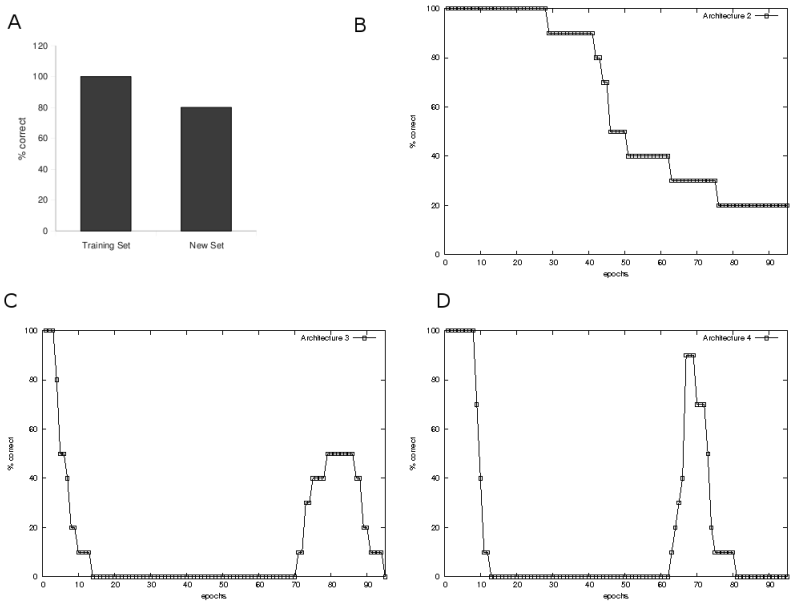

Fig. 3 The simulation results. A) The results with the Feed-Forward Architecture for Classification. 100% of the training samples and 80% of the test samples were recognised; B) The results with the Architecture with Recurrent A-SOM Connections. After ceasing to receive any input this architecture was able to continue to produce 100% correct sequences of output for 28 epochs, and then continue with 90% correct sequences until epoch 42; C) The results with the Architecture with Recurrent Connections from the Output Layer to the A-SOM. There was 100% correct reproduction of the output sequence in the first 3 epochs after ceasing to receive any input; D) The results with the Architecture with Recurrent A-SOM Connections and Recurrent Connections from the Output Layer to the A-SOM. There was 100% correct reproduction of the output sequence in the first 8 epochs after ceasing to receive any input.

3.4 Architecture with Recurrent A-SOM Connections and Recurrent Connections from the Output Layer to the A-SOM

The simulation results with the fourth architecture are depicted in Fig 3 D. In this figure we can see that this architecture was able to reproduce the sequence of the training samples with 100% correctness in the first 8 epochs (i.e. for 80 iterations), then there was a rapid decline until a level of 0% correct activities were reached at epoch 13. This level of 0% correct activities continued until epoch 63 when it started to improve again. It reached a new peak of 90% correct activities between epochs 67 and 69. After that there was a decline again and the level of 0% correct activities was reached again at epoch 81. This level of 0% correct activities was still the performance level at epoch 95.

4 Discussion

We have implemented and tested four supervised A-SOM based architectures. The first architecture was a Feed-Forward Architecture for Classification and it was able to correctly classify 100% of the training samples as well as 80% of a new set of test samples. The three other architectures used recurrent connections to enable internal simulation of perceptions and actions.

The second architecture used recurrent connections to feed back the activity of the A-SOM to itself as ancillary activity with a time delay of one iteration. After ceasing to receive any input this architecture was able to continue to produce 100% correct sequences of output for 28 epochs, and then continue with 90% correct sequences until epoch 42.

The third architecture used recurrent connections to feed back the activity of the output layer to the A-SOM as ancillary input with a time delay of one iteration. This architecture was able to continue to produce 100% correct sequences of output for 3 epochs after ceasing to receive any input.

The fourth architecture used two sets of recurrent connections. One set of recurrent connections were used to feed back the activity of the A-SOM to itself as ancillary activity with a time delay of one iteration. The other set of recurrent connections were used to feed back the activity of the output layer to the A-SOM as ancillary input, also with a time delay of one iteration. This architecture was able to continue to produce 100% correct sequences of output for 8 epochs after ceasing to receive any input.

When comparing the three architectures with recurrent connections, the architecture with recurrent connections that feed back the activity of the A-SOM to itself is clearly the best. This is because it is able to continue with proper output sequences much longer than the other recurrent architectures in the absence of input.

That the correctness of the output sequences decline with time in the three architectures with recurrent connections is reasonable and it is probably due to that the

present activity elicit similar (but not exactly the same) activity in the succeeding iteration, which over time should lead to an increased deviation from the correct activity.

A reasonable guess to why the architecture with recurrent connections that feed back the activity of the A-SOM to itself is better than the architecture with recurrent connections from the output layer to the A-SOM is as follows: It should be possible to keep more information when associating the activity in the A-SOM with the activity of the A-SOM in the previous iteration than when associating the activity of the A-SOM with the activity of the output layer in the previous iteration. The reason is dimensionality, i.e. the number of connections is much larger in the former case than in the latter because the A-SOM in our simulations contains 225 neurons whereas the output layer contains only 10.

The reason that there is a second peak in the two architectures with recurrent connections from the output layer to the A-SOM is probably similar. There is a higher probability that the activity pattern in the 10 output neurons starts to become proper again after some time than that the activity pattern in the 225 neurons in the A-SOM happens to become proper. However, we have no idea about why the second peak comes earlier, reaches a higher level and is more narrow in the fourth architecture than the second peak in the third architecture.

A somewhat surprising result was that the fourth architecture, i.e. the one with recurrent connections from the output layer to the A-SOM as well as recurrent connections from the A-SOM to the A-SOM is not better than the second architecture, i.e. the one with recurrent connections from the A-SOM to itself. We had expected it to be because it should be able to keep more information. Probably this is due to the way the total activity is calculated in the A-SOM, i.e. by averaging the ancillary activities and the main activity. This means that the ancillary input from the output layer will have as much influence on the total activity of the A-SOM as the ancillary activity with the time delayed A-SOM activity. Thus a performance somewhere in between the performance of the architecture with recurrent connections from the A-SOM and the performance of the architecture with recurrent connections from the output layer to the A-SOM would be expected. This is also what we got in the simulations.

One idea for further development of the presented architectures is to use several sets of recurrent connections with different time delays to improve the ability to continue with proper output sequences in the absence of input. One drawback with this approach is of course the increased computational burden. Thus it will be a matter of weighting the improved ability to continue with proper output sequences against the additional computational burden.

Another idea is that it is conceivable to develop a variant of the A-SOM based on the Growing Cell Structure [4] or the Growing Grid [5]. In this way it might be possible to create an architecture that automatically creates a suitable number of neurons with a suitable topology. This would yield a suitable size of the hidden layer to represent the clusters in the particular input space.

References

[1] C. Balkenius, J. Morén, B. Johansson, and M. Johnsson. Ikaros: Building cognitive models for robots. *Advanced Engineering Informatics*, [doi:10.1016/j.aei.2009.08.003], 2009.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[4] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7, 9:1441–1460, 1993.

[5] B. Fritzke. Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2, 5, 1995.

[6] G. Hesslow. Conscious thought as simulation of behaviour and perception. *TRENDS in Cognitive Sciences*, 6:242–247, 2002.

[7] M. Johnsson and C. Balkenius. Associating som representations of haptic submodalities. In S. Ramamoorthy and G. M. Hayes, editors, *Towards Autonomous Robotic Systems 2008*, pages 124–129, 2008.

[8] M. Johnsson and C. Balkenius. Experiments with self-organizing systems for texture and hardness perception. *Global Journal of Computer Science and Technology*, 1.2:53–62, 2009.

[9] M. Johnsson, C. Balkenius, and G. Hesslow. Associative self-organizing map. In *International Joint Conference on Computational Intelligence (IJCCI) 2009*, pages 363–370, 2009.

[10] M. Johnsson, D. Gil, C. Balkenius, and G. Hesslow. Internal simulation of perceptions in a bimodal system. *(Manuscript)*, 2009.

[11] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag, 1988.

[12] T. Ziemke, D Jirenhed, and G. Hesslow. Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68:85–104, 2005.