Bugmann G. (2009) Proceeding of IICAI'09, Tumkur (Bangalore), pp 828-841

A Neural Architecture for Fast Learning of Stimulus-Response Associations.

Guido Bugmann,

School of Computing and Mathematics
University of Plymouth, Drake circus, Plymouth PL4 8AA, United Kingdom.
gbugmann@plymouth.ac.uk

Abstract. Humans can learn in a matter of seconds to associate a response R with a stimulus S, following simple verbal instructions, e.g. "Press the right button when you see a green light". This involves establishing links through several neural relays, from visual to motor representations. There is no model of fast learning in such systems. This paper describes a new architecture and learning principle that enable fast associative learning. Learning proceeds through an "activation pull" from the response area and a "representational push" from the stimulus area. This novel pull-push principle allows separated pattern recognition streams to develop in the network upon single presentations of S-R pairs. A proof-of-concept implementation is presented in the paper in which arbitrary *n x n* binary stimuli are mapped to arbitrary *n x n* responses (*n* =10 and 15). The results and the capacity of the network are discussed.

Keywords: Neural networks, Associative learning, Boolean function, Brain, Stimulus-Response learning.

1 Introduction

Humans are routinely "programmed" through verbal or written instructions. Take the example of a new employee to whom the job is being explained, or a trainee driver who is told: "Stop when the light is red". Before every experiment investigating human capabilities, subjects are programmed (told) to perform in the required manner. This ability of humans to convert instructions into response sets is one of the most fascinating of the human capabilities. Yet, very little brain research has been devoted to it. To the best of our knowledge, no modeling work has attempted to explain how the brain sets up the internal program to perform a task on the basis of instructions. No brain scan data have ever been reported on the instruction phase of subjects asked to produce motor responses to stimuli. The nearest experiments focused on subjects receiving step by step instructions to mentally assemble objects [1][2].

These experiments show that instructions activate both the premotor cortex and the parietal cortex, two relays on the most direct path from visual sensory input and motor output [3]. Listening to action descriptions has the same effect [4]. Other experiments

show that action verbs activate areas of the premotor cortex which are specific for the action described by the verb, e.g. face or leg actions [5]. Imagining an object activates several stages of the visual information processing stream, down to the primary visual area V1, the first cortical stage of image processing [6]. It appears that instructions can generate neural images of the stimulus and the response along the same path that the actual stimulus will use to elicit the actual responses. In [7] it was noted that areas of the parietal cortex became responsive to the stimulus only after it became relevant for the task. The simplest explanation is that SR learning is a modification of the visuo-motor path guided by neural images generated in imagery mode as a consequence of instructions.

This learning process is not understood. We know that it is fast, as it takes only a few seconds to prepare for a verbally instructed stimulus-response association. This excludes learning through the creation of new synaptic connections. A stimulus can occupy only a small part of the visual field that can be completely represented at a lower stage of the visual system. Thus, the SR learning process must be able to create links between activity patterns separated by several synaptic relays. In [3], as study of visuo-motor coordinate transformation leads to the suggestion that the transformation is not localized in a specialized area, but is progressively built as the activity "percolates" along the visuo-motor path.

There are many neural network models able to learn input-output associations, but none has the desired property of fast learning from the presentation of a single S-R pair. For instance, the Multi Layer Perceptron needs repeated presentations of all S-R pair, otherwise it undergoes "catastrophic forgetting" [8]. The same holds for networks based on the associative paradigms of Hopfield or Willshaw. See for instance in [9] a study of a problem similar to the one addressed in this paper. In approaches of the reservoir family, e.g. [10], the representation capabilities are fixed and cannot adapt to arbitrary patterns. Radial-Basis Function networks can learn rapidly and flexibly [11] but require an implausible large fan-in and fan-out connectivity. They do not offer a solution to the problem of developing associations across several neural relays.

In section 2, a new neural network concept and design is proposed, that offers a solution to the above problems. In section 3, the learning algorithm is detailed. In section 4, the theoretical capacity of the network is estimated. In section 5, results of fast association learning are shown and evaluated. In section 6, network properties and open issues are discussed. Section 7 is the conclusion.

2 Proof of concept model

2.1 Model concept

The new network model was developed to propose at least one solution to the problem of fast associative learning. Issues of strict biological plausibility and optimality are left for future work (see discussion). A S-R association model should allow fast, one shot learning, use reasonable fan-in and fan-out connectivity, allow mapping between any input stimulus and any output response across several layers,

and learn new S-R associations without affecting previously learnt ones. Following concepts guided the design of the solution. Implementation details are given in section 2.2.

A. It is assumed that the functionality of high fan-in and fan-out nodes can be emulated by a multilayer architecture of relay nodes with low fan-in and fan-out. The problem of S-R mapping is approached here as a problem of finding a path through a network, to link the input layer, representing the stimulus, and the output layer, representing the response.

B. To produce a response that is specific to a given stimulus, each node in the selected path will act as a recognizer of the pattern of inputs in its receptive field (RF). By cascading such local recognition functions, nodes in the last layer will be able to represent complete stimuli. The pyramidal connectivity and number of layers is designed such that any output neuron is able to receive input from all parts of the input layer.

C. To develop such cascades of recognizers, learning uses an "activity pull" and a "representation push" (Figure 1). The activity pull is information sent back from the output layer towards the input layer. For instance, if a specific node in the output needs to be activated when the stimulus is presented, this node will send an "enabling" signal back to all nodes in previous layers that can feed this output node. This creates a "cone" of enabled neurons through which activity must necessarily flow to reach the specific output. Through the design of the architecture, the cone necessarily includes the full input layer. The representation push is a mechanism which selects, in the enabled part of each layer, a minimum number of neurons necessary to completely represent their input pattern. These neurons form a "representation tree" (the tree-growth procedure is described in section 3). The active neuron in the last layer becomes a detector for the stimulus currently presented.

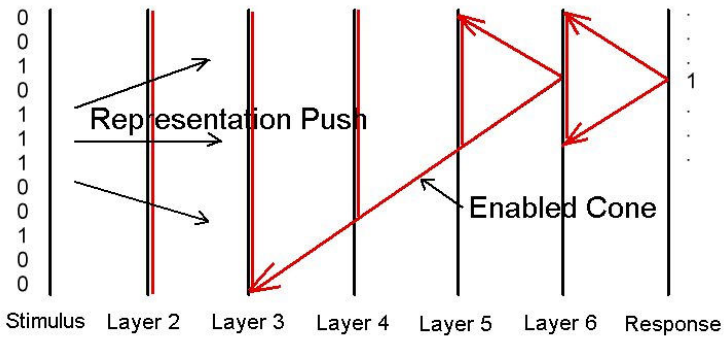


Fig. 1. Illustration of the learning principle. A node in layer 6, corresponding to an area of desired activity in the Response layer, emits an "activity-pull" signal that propagates back through the network of layers, creating a cone of "enabled" neurons. A forward push is then generated by the stimulus aimed at recruiting nodes in the cone that are able to represent the stimulus. The connectivity of the architecture ensures that a single node in layer 6 will become a specific detector of the full stimulus.

2.2 Model implementation

The model has the following properties:

- The model operates with binary Stimulus images and Response patterns. The training set consists of 26 pairs of images with capital letters being the stimulus, and lower-case letters representing the response, e.g. A-a, B-b, etc. These were centred in 10x10 or 15x15 pixel images. Pixels/input nodes belonging to a letter have a value of 1 and those belonging to the background have a value of 0.

- The network has 7 layers, each of comprising 10x10 nodes. Experiments were also conducted with 15x15 node layers. Layer 1 is the Stimulus and layer 7 is the Response layer. Each node has a 5x5 receptive field in the previous layer (7x7 in the 15x15 case). This ensures that any neuron in layer 6 can receive input from any neuron in layer 2. Layers 2 to 6 are representation layers.

- All weights are either 1 or 0. The first representation layer also uses negative input weights of -1 from inputs that are not active in the stimulus.

- Nodes are binary neurons with an adjustable threshold. They become active (output = 1) only if the weighted sum of the input exceeds the threshold. In this model, where, at the start of learning, all weights are set to 1, the weighted sum of inputs is the count of active inputs in the neuron's receptive field, and the threshold can be seen as a specifying the minimum number of active inputs.

- Nodes have two state parameters: enabled_state and recruited_state. They are either enabled or disabled and either recruited or available. An enabled neuron can be activated by its inputs and can learn to recognize its inputs' configuration. Only neurons in an activity-pull cone are enabled. A recruited neuron has learnt a particular input configuration, i.e. its weights with only certain input neurons are non-zero, and it will only respond to that configuration. A recruited neuron is trained to recognize a certain local feature of a specific stimulus, but can be activated by any other stimulus that has the same local feature and can contribute to the representation tree of that other stimulus.

- Number of required layers: For a neuron in the layer before the Response to see any neuron in layer 1 (Stimulus), given a size of the receptive field (RF) of N_{RF}, requires the following number of layers L before the Response layer:

$$L = 1 + \frac{\sqrt{N} - 1}{0.5\sqrt{N_{RF}} - 0.5} \qquad (1)$$

  (to be rounded up to nearest integer) e.g. for N=100 and N_{RF} = 25, we find L= 6.

- Number of connections: A simple three-layer network with equivalent functionality uses N RBF nodes in the hidden layer with N inputs from the Stimulus layer and N outputs to the Response layer. The number of connections in such a network is N_{CRBF}:

$$N_{CRBF} = 2N^2 \tag{2}$$

The proposed L-layer network has in each layer N nodes with N_{RF} connections. Its total number of connections N_{CL} is:

$$N_{CL} = NN_{RF}(L-1) \tag{3}$$

For a 10x10 input and output N_{CRBF} = 20'000 and for a 7-layer network, N_{C7} = 15'000. For a 15 x 15 case, N_{CRBF} = 100'000 and N_{C7} = 66'000

The proposed multilayer architecture requires less connections than the RBF network, but there is also a significant difference in capacity. The RBF network can learn exactly N different S-R pairs, while the 7-layer network can only achieve this in a very special case (see section 4).

3 Training

For each S-R example, training has following stages

1. Target node selection in the Response layer
2. Setting up the activity-call cone centred on the target node.
3. Building a representation of the stimulus using nodes in the cone.
4. Associate as many Response nodes as possible to the built representation.
5. If response nodes are not all activated, return to stage 1 and setup a new cone centered on inactivated nodes in the Response.

3.1 Target node selection

When a training S-R pattern is loaded, the stimulus image S is copied into the Stimulus layer, with nodes corresponding to letter pixels being set to 1 and nodes corresponding to the background being set to 0. The image of the Response is stored in a "target" buffer of the Response layer. This normally contains the "desired output" of the Response layer.

An active pixel of the desired response is selected that is close to the middle of the pattern. This choice maximizes the number of Response pixels that will be associated with a developed representation node in layer 6. In practice, a list of active nodes is produced and the node closest to the middle of the list becomes the target, i.e. the training procedure will attempt to create an association between the stimulus and the nodes in the neighbourhood of that node. Once a representation tree has been constructed, a number of Response nodes within range of the top of the tree are associated with the stimulus and are removed from the list. In the next round, this then naturally leads to another target node to be selected.

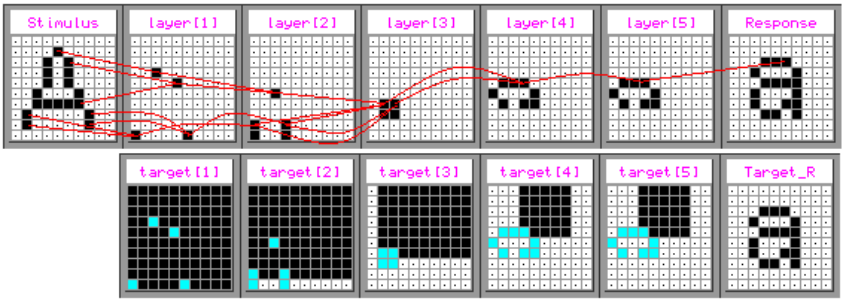


Fig. 2. Example of representation tree built for the A-a association. The upper row of grids shows the activity of nodes in the 7 layers arising from the presentation of the stimulus "A". A black node has an activity of 1. A white node has a zero activity. The (red) lines linking the nodes show the representation tree built for the upper-right active node in layer 6 (labeled "layer[5]" in the simulation). This node activates a single node in the Response layer. The lower row of grids is used for displaying node parameters. The blue/light-grey nodes are those which have been recruited during the learning of the "A-a" association. Only four nodes in layer 2 (labeled "Layer[1]") were needed to fully represent the stimulus "A". The black nodes indicated the enabled nodes for the development of a tree towards activating the top-right active node of the "a" in the Response.

3.2 Setting up the activity-call cone

Each node has an enabled_status parameter that signals if the node is enabled for learning and activation, or disabled. Each node in layer 6 that projects to the target node in the Response layer is now set to be enabled. The target node in the Response layer has a corresponding node in layer 6, at the same position in the grid. This node becomes the actual head of the enabled cone. Every node in layer 5 that projects to this one becomes enabled. Then, every node in layer 4 that projects to an enabled node in layer 5, is also enabled. This continues all the way down to layer 2. Figure 2 shows an enabled cone. All nodes that have previously been recruited can also be enabled, so that they can become active during a stimulus presentation. However, they cannot be re-trained.

3.3 Building a representation of the stimulus

This is the representational push and it starts in layer 2. All nodes are initialized with weights equal to 1 and with a maximum value of their threshold. The maximum value is set to N_{RF} + 1. This ensures that, even if all N_{RF} inputs were active, the neuron would not fire. Then, the threshold is progressively reduced for all enabled and not yet recruited nodes. At some threshold value, the node with most active inputs will become active. For this node, the threshold is frozen and its recruited_status parameter is set to "recruited". The weights from the Stimulus pixels to that node are left at 1 if the pixel is active, and are set to -1 if the pixel is inactive and has not been active as part of the full stimulus. Weights from parts of the stimulus that are normally active are set to 0. This implements a form of push-pull model of the

receptive field of visual neurons in V1 [12], Then, all active inputs of that node are inhibited. Thereby, a subset of active pixels in the stimulus is removed and no other node in layer 2 will attempt to represent that part of the stimulus. The threshold lowering process continues until no activity is left in the Stimulus. At that point one can be certain that every pixel of the stimulus has been represented in layer 2. Figure 2 shows that 4 neurons in layer 2 were required to represent the stimulus "A". The representational push now moves to layer 3, with the aim of recruiting the necessary number of nodes to represent activity in layer 2. Here, the inputs weights are set in a more simple way to 1 for active inputs a 0 for inactive inputs. The shape of the cone is such that the process ends with a single node recruited in layer 6, which represents the whole stimulus. An example of the built representation tree is shown in figure 2. In principle, elements of trees built for previous patterns can be integrated in a new tree, as recruited nodes are allowed to become active. Their presence is tested at the beginning of the representational push in each layer, by updating every node in the activity-call cone. It is possible that, in a trained network, no free enabled node can be found to represent parts of the activity in the previous layer. This prevents the full stimulus to be represented and can lead to the recruitment of nodes in the last representation layer (layer 6) that only respond to a part of the stimulus. Such nodes cause interferences between learnt patterns. To prevent their creation, the existence of un-extinguished nodes in the previous layer is examined. If these are found, training is aborted for the current cone and a new tree head is sought. For some Stimulus patterns, it can be impossible to find any node to build a representation tree, and the S-R remains un-associated.

3.4 Association with Response nodes

All weights between layer 6 and the Response layer are set to 0. All thresholds in the Response layer are set to 0. The one top-of-the-tree node recruited in layer 6 is necessarily located in the RF of the target node in the Response layer. All active nodes of the desired response with input connections from the top-of-the-tree node are now associated to that node, by setting the connection weight to 1. From now on, each time that the trained node in layer 6 fires, it will also activate a number of pixels in the Response pattern. These pixels are then extinguished from the desired response, and cannot exert an activity pull anymore. The same pixel can be activated as part of several S-R associations. The threshold of zero ensures that the response pixel can be activated by a single active connected neuron in layer 6. In this implementation, Response neurons become OR gates. The interferences between different S-R pairs is avoided by the selectivity of neurons in layer 6[1].

[1] Input patterns never seen during training do not elicit any response in layer 6. They can elicit activity in lower representation layers if they share local features with a trained stimulus.

4 Capacity

Layer 6 has N neurons and can theoretically represent N different stimuli. There is however a limit of the number of trees that can be formed using nodes in representation layers. This limit depends on the degree of sharing of representation nodes between stimuli and on the number of trees needed for generating a response pattern. We evaluate here another limit that arises from the connectivity between layer 6 and the Response layer. The actual capacity depends on the distribution of active nodes in the Response pattern.

a. In the common case, where different Responses are sharing active nodes, then, the limiting factor is the number of inputs N_{RF}. Each Response neuron can only see N_{RF} nodes in layer 6 (N_{RF} = 5x5 = 25 for the 10x10 case, and 7x7 = 49 for the 15x15 case). Thus, each Response neuron can, at most be activated by N_{RF} different stimuli. This only applies to neurons in Response placed at some distance from the boundaries. Neurons in the corner would only receive inputs from $N_{RF}/4$ inputs. So, if all response patterns include active nodes in the corners, the maximum capacity drops to $N_{RF}/4$. Thus the maximal capacity, as limited by RF size of Response neurons, is C_{NRF}:

$$N_{RF} / 4 < C_{NRF} < N_{RF} \tag{4}$$

In the 10x10 case, the capacity C_{NRF} lies between 6 and 25 S-R associations and, in the 15x15 case, it lies between 12 and 49.

b. In the extreme case where each response comprises only one active node (different for each response), the network could associate N stimuli to their responses. The theoretical maximal capacity C_{MAX} is then:

$$C_{MAX} = N \tag{5}$$

c. In the other extreme, where each response pattern has active neurons evenly distributed over the Response layer, the network needs to generate several active nodes in layer 6, each fully representing the stimulus. The reason is that each active node in layer 6 can only feed N_{RF} nodes in the Response. The number of active nodes needed in layer 6 is the number of adjacent RFs that can be placed in layer 6.

$$N_{needed} = (1 + \frac{\sqrt{N}}{\sqrt{N_{RF}}})^2 \tag{6}$$

e.g. in both our cases: Nneeded = 9

In such a case, one would not be able to represent more than C_{MIN} different patterns in layer 6:

$$C_{MIN} = \frac{N}{N_{needed}} \tag{7}$$

In the 10x10 case: $C_{MIN} = 11$ and in the 15x15 case: $C_{MIN} = 25$.

The training data used in this paper are closest to cases a and c, with extended patterns centred in the Response layer and do not include pixels near the corners. The capacity is therefore expected to lie between C_{MIN} and N_{RF}, i.e. between 11 and 25 in the 10x10 case, and between 25 and 49 in the 15x15 case.

One can now estimate the connections cost of representing S-R associations. Using the minimum capacity C_{MIN} (7) and the number of connections (2) and (3), one finds that the 10x10 net used approximately 1400 connections per S-R association (200 connections for the 10x10 RBF net), the 15x15 net uses 2600 connections (450 connections for the 15x15 RBF net). So, the proposed multilayer architecture is more costly in terms of number of connections per S-R associations. Another consideration however is the fan-in. Each RBF nodes must receive inputs from all pixels in the stimulus, while the nodes in the multilayer network only receive N_{RF} inputs. The latter can be reduced by increasing the number of layers.

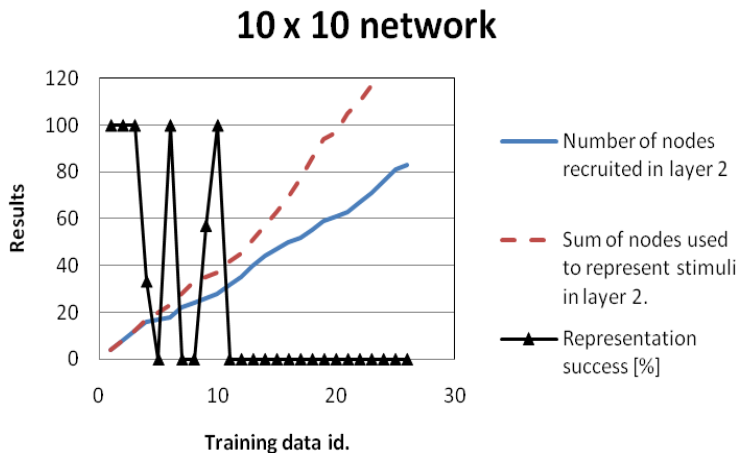
5 Results

Following experiments were conducted. Two networks were used: A 7-layer 10x10-node net and a 7-layer 15x15-node net. The S-R pairs were the same in both cases and each pair was presented only once in alphabetic order.

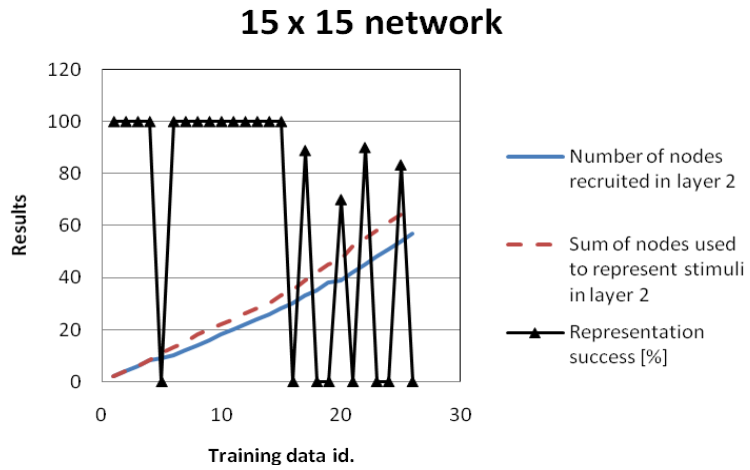
For each S-R pair, the representation success was measured. The representation success is the fraction of correct pixels in the response that the net is able to learn to associate. The fraction of correct pixels is calculated for all pixels that have a value of 1 either in the generated response or in the desired response. Restricting the measurement to pixels that are or should have a value of 1 eliminates background pixels from the calculation and gives a measure of success that feels correct to a human observer of the results.

Figure 3 shows the performance for the 10x10 and 15x15 nets:

The 15x15 net is able to learn perfectly 14 out of the 26 S-R associations. Another 4 have a representation success above 70%. The 10x10 net represented perfectly 5 associations. Another 2 have a representation success between 30% and 60%. Both experimental capacities nets are well below the theoretical minima of 25 for the 15x15 case, and 11 for the 10x10 case, calculated in section 4.



(A)



(B)

Fig. 3. Results of learning experiments. Training data for 26 stimulus-response (S-R) associations were presented to the network. Data 1 corresponds to the "A-a" association, data 2 corresponds to "B-b", and so on, until data 26 which corresponds to the "Z-z" association. (A) Results for training a network with 10x10 layers. The triangle symbols indicate the representation success for each S-R set. Less than 100% indicates that only parts of the response were associated with the stimulus. The full (blue) line shows the total number of nodes recruited in layer 2. The dotted line shows the accumulated number of nodes used for each S-R learning in layer 2, irrespective of if the nodes used were previously trained nodes or newly recruited ones. (B) Same results for a network with 15x15 layers. The size of the letters in the S-R pairs were the same in both cases. Larger images were produced for the 15x15 case by adding rows of pixels around the images used for the 10x10 case.

The main cause for non associations is the lack of free nodes to build a new tree within the specified cone. This problem becomes the more severe the more

associations have been learnt. The 10x10 network is running out of capacity much earlier than the 15x15 net.

It is possible that the true limiting factor to the capacity is the one ignored in the calculation in section 4, namely the recruitment of nodes during tree growth. Other points worth considering are: i) the nature of S-R images used here could stress resources and ii) the implementation may not be making the best use of available nodes, e.g. concentrating all nodes recruitment in the same area, thus causing an early depletion of resources.

Nodes in layer 2 are image feature detectors built as learning progresses. As these detectors can be shared for different associations, one can expect a saturation of the number of new detectors created as the network is exposed to more stimuli. This is not observed in this experiment. Figure 3 shows a relatively linear relation between number of detectors developed in layer 2 and number of trained S-R pairs. The main reason is that the receptive fields are large enough to cover a significant part of the stimulus, and therefore become specialized for relatively complex and stimulus-specific patterns of pixels. Nevertheless, there is some evidence of detector sharing. The average number of nodes recruited per stimulus in layer 2 in the 15x15 case is 2.2. The average number of nodes used to represent a stimulus is 2.6. Therefore, few recruited nodes have been shared in this case. In the 10x10 case, the average number of recruited nodes is 3.2, while the average number of nodes used is 5.2. Thus, much more sharing is taking place in the 10x10 case. This is likely to be due to the smaller receptive fields in the 10x10 case (5x5 vs 7x7).

Regarding computation load, the learning algorithm requires one backward pass and one forward pass per tree. Each SR association may require several trees, where each tree generates a part of the response pattern. For instance, the A-a association in figure 2 requires 8 trees (this could be reduced with a smarter selection of tree heads in section 3.1). Therefore, a "one-shot" learning actually requires 8 forward and 8 backward passes through the network.

6 Discussion

The proposed learning algorithm uses an unusual approach with all input weights being initially in an ON state (weights=1), ready to transmit any activity of their inputs. The node most suitable to represent that input activity is then found through a progressive reduction of the threshold. Once a node starts responding, the weights are then adapted in a standard Hebbian fashion, with weights encoding the joint pre- and post-synaptic activity. Functionally, this is equivalent to starting learning with a high global inhibition and then slowly reducing it until the node with the most input starts firing. A biological implementation of the proposed paradigm would need a global control signal of the excitability of neurons in a layer.

A key element of the algorithm is the implementation of an enabled cone that channels any representation tree development towards a location suitable for response generation. One can imagine that one area of the Response layer controls arm movements while another one controls leg movements. Setting up the cone requires information flowing backwards from the area of desired output to all areas able to

feed this area through multiple synaptic relays. It may be worth investigating if some of the multiple feedback connections found in the brain could carry enabling signals.

The proposed approach does not require new synaptic connections to be generated. This is one of the pre-requisites of a biologically plausible fast algorithm. In the representation layers, the algorithm actually operates by making existing weights inactive and a trained tree ends up with only a small fraction of active weights. In the Response layer, the opposite takes place, with initially inactive weights later becoming active. There may be a relation to increases and decreases of the fraction of so called "silent synapses" in different parts of the developing biological visual system [13].

Layers 2 to 6 form a cascade of AND(N_{RF}) functions over the N_{RF} inputs of their receptive fields. These 5 layers constitute an equivalent AND(N) function over all N nodes of the input layer 1. Nodes in the output layer realize an OR(N_{RF}) function (figure 4). Replacing the cascade with one layer of AND(N) nodes would use more connections than the cascade architecture but is more cost-effective in terms of number of connections per learnt S-R association. Biological systems do not have the option of using large fan-in neurons covering the whole visual field and the cascade could be a good alternative solution. The biggest technical advantage of the cascade is that it uses a smaller number of local connections per node, facilitating the design of dedicated hardware. In the extreme, the cascade could make use of nearest-neighbour connections only, similarly to those found in cellular automata. This would indeed require a large number of layers. A larger fan-in would still be required from the Response layer to maintain capacity, unless an appropriately sparse coding of the response can be designed.

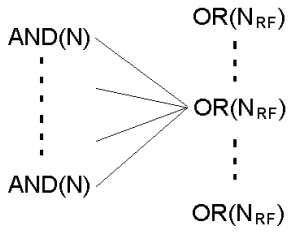


Fig. 4. Equivalent Boolean function of the trained network. Each AND(N) function represents one trained recognition tree. N is the number of nodes in the Stimulus layer. Each OR(N_{RF}) function represents a node in the Response layer. N_{RF} is the number of inputs in the receptive field of each node.

The first representation layer implements a form of push-pull receptive field organization [12], where layer 2 neurons are locally (i.e. within their receptive field RF) activated by the stimulus S_{RF} and are inhibited by the negative image of the stimulus $\overline{S_{RF}}$. This is not equivalent to a push-pull organization of the equivalent AND(N) over the whole input stimulus S. For instance, the local inhibitory field for the stimulus $\boxed{1}$ might not inhibit the zero of the stimulus $\boxed{1\ 0}$, if the zero is spatially too far from the 1. Thereby, the response to the 1 and 10 may be both activated by the

presentation of the stimulus 10. It is presently unclear how to modify the algorithm to implement an equivalent push-pull over the full S and $\overline{S}$ inputs.

This work focused on the association element of learning S-R mappings. Other elements are needed for building a full model of learning from verbal instructions. This includes two imagery components, converting verbal input into mental images of the stimulus and of the motor response respectively. It is tempting to investigate if associative networks similar to the one presented here might be usable for these tasks. It is also worth investigating if and how the capacity can be increased to become of practical use as a memory for life-long learning. Alternatively, this network might be best suited for a small number of continuously renewable short-term associations.

7 Conclusion

This paper has introduced a new one-shot learning paradigm enabling the development of recognition trees across several layers. The activity-pull-representation-push learning principle has shown its effectiveness. Initial experiments suggest that the capacity is mainly limited by the depletion of neural resources used to build recognition trees. The work was motivated by the lack of model of rapid setup of S-R association by humans. The proposed model needs further development of its biological plausibility before it can be considered as a model of human S-R learning. The presented proof-of-concept results give hope, if not guidance, for the development of such a model. On the AI technology side, the work has shown that large fan-in AND(N) functions can be replaced with a more tractable tree of small fan-in AND(N_{RF}) functions that can be generated automatically from training data.

Acknowledgments.

The author is grateful for comments by Thomas Wennekers on an earlier version of this paper.

References

1. Mellet, E., Tzourio, N., Crivello, F., Joliot, M., Denis, M., Mazoyer, B.: Functional anatomy of spatial mental imagery generated from verbal instructions. J Neurosci 16: 6504-6512 (1996).
2. Sack AT, Jacobs C, De Martino F, Staeren N, Goebel R, Formisano E.: Dynamic premotor-to-parietal interactions during spatial imagery. Journal of Neuroscience, 28, 8417-8429 (2008).
3. Caminiti R.: From vision to movement: Combinatorial Computation in the Dorsal Stream. In: Caminiti R., Hoffmann K.P., Lacquaniti F. and Altmann J. "Vision and Movement Mechanisms in the Cerebral Cortex", HFSP, Strassbourg pp. 42-49 (1996).

4. Tettamanti M., Buccino G., Saccuman M.C., Gallese V., Danna M., Scifo P., Fazio F., Rizzolatti G., Cappa S.F., Perani D.: Listening to action related sentences activates fronto-parietal motor circuits. J. Cogn Neurosci, 17:273-281 (2005).
5. Pulvermüller, F., Härle, M., Hummel, F.: Walking or talking?: Behavioral and electrophysiological correlates of action verb processing. Brain and Language, 78, 143-168 (2001).
6. Klein, I., Paradis, A.-L., Poline, J.-B., Kosslyn, S. M. and LeBihan, D.: Transient activity in human calcarine cortex during visual imagery. Journal of Cognitive Neuroscience, 12, 15S-23S. (2000).
7. Tosoni A, Galati G, Romani GL, Corbetta M.: Sensory-motor mechanisms in human parietal cortex underlie arbitrary visual decisions. Nature Neuroscience 11, 1446 - 1453 (2008).
8. McCloskey M. and Cohen N.: Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (ed.) The Psychology of Learning and Motivation: Vol. 24, 109-164, NY: Academic Press (1989).
9. Garagnani, M.; Wennekers, T.; Pulvermuller, F.: A neuroanatomically grounded Hebbian-learning model of attention-language interactions in the human brain. European Journal of Neuroscience 27, 492-513 (2008).
10. Morse, A. F. & Aktius, M.: Dynamic Liquid Association: Complex Learning Without Implausible Guidance. *Neural Networks, in press (2009)*.
11. Bugmann G.: Role of the cerebellum in time-critical goal-oriented behaviour: Anatomical basis and control principle." in Wermter S., Austin J. and Willshaw D. (eds) "Emergent Neural Computational Architectures Based on Neuroscience", Lecture Notes in Computer Science (LNAI 2036), Springer, pp. 255-269 (2001).
12. Ferster, D., and Miller K.D.: Neural mechanisms of orientation selectivity in the visual cortex, Ann. Rev. Neurosci., 23:441-471 (2000).
13. Rumpel S., Kattenstroth G. and Gottmann K.: Silent Synapses in the Immature Visual Cortex: Layer-Specific Developmental Regulation. J Neurophysiol 91: 1097-1101 (2004).