LEARNING TO GENERATE ARTIFICIAL FOVEA TRAJECTORIES FOR TARGET DETECTION

(*International Journal of Neural Systems*, 2(1 & 2):135–141, 1991. Figures omitted!

)

JÜRGEN SCHMIDHUBER [*], TUM
RUDOLF HUBER, TUM

Abstract

It is shown how 'static' neural approaches to adaptive target detection can be replaced by a more efficient and more sequential alternative. The latter is inspired by the observation that biological systems employ sequential eye-movements for pattern recognition. A system is described which builds an adaptive model of the time-varying inputs of an artificial fovea controlled by an adaptive neural controller. The controller uses the adaptive model for *learning* the sequential generation of fovea trajectories causing the fovea to move to a target in a visual scene. The system also learns to track moving targets. *No teacher* provides the desired activations of 'eye-muscles' at various times. The only goal information is the shape of the target. Since the task is a 'reward-only-at-goal' task , it involves a complex temporal credit assignment problem. Some implications for adaptive attentive systems in general are discussed.

1 INTRODUCTION

We study an aspect of adaptive vision with neural networks which has not been explored in this general form before: The *adaptive* control of sequential physical fovea-movements for target detection.

Consider the following target detection task: A two-dimensional object may be arbitrarily rotated and translated on a pixel plane consisting of many pixels. Learn to give the position and the orientation of a predefined detail of the object (the target).

[*]Research supported by a scholarship from SIEMENS AG

Now consider the *naive* 'neural' solution to this task: By providing a huge number of training examples, train a feed-forward network with many input units (typically one for each pixel), many hidden units and many (typically millions of) connections to emit a representation of the position and the orientation of the target.

The contribution of this paper is a system for target detection which can be more efficient, more sequential, but also more complex than the naive approach. It is inspired by the observation that biological systems employ sequential fovea movements for target detection. The system is capable of *'active perception'*: At a given time it can have an influence on what to perceive next. It learns to produce sequences of fovea movements (rotations and translations) which lead the high-resolution part of an artificial fovea from arbitrary starting points in the environment of a randomly placed object to a predefined detail of the object (the externally defined target). In particular, we show how techiques for adaptive neuro-control can be used for learning target detection without an informed teacher (the task is a *'reward-only-at-goal'* task). The system solves its target detection task solely by being given the shape of the target, but *without being told how to get there.* It *learns* to focus on those domain-dependent parts of the visual scene which are relevant for the target detection process. The system is efficient in the sense that it uses only a fraction of the input units and connections of the naive approach, still allowing maximal resolution to be applied to each part of the pixel plane.

The remainder of this paper is structured as follows: First we describe and motivate our 2-network approach for solving the *temporal* credit assignment problem associated with the target detection task.

Then experiments with target detection problems are described. It is demonstrated that the system can discover (in an unsupervised manner) target-directed trajectories (sequences of fovea translations and rotations) by learning to sequentially focus on relevant cues in the visual scene. As a by-product, the system *learns* translation and rotation invariance, as well as target tracking. It is demonstrated that an *imperfect* adaptive model of the environmental dynamics can contribute to *perfect* solutions. It is also demonstrated that making a sequential task out of a static one can be very efficient. Furthermore, a method for parallel on-line learning of both networks is experimentally shown to be feasible.

Finally implications for more general attentive systems are discussed.

2 THE SYSTEM

Subsection 2.1 gives the rationale behind our system. Subsection 2.2 provides the formal details.

2.1 Outline and Motivation of the System

There is an artificial movable fovea with predefined 'receptive fields'. At the beginning of a target detection process the fovea is placed somewhere on a pixel plane. The artificial fovea is coarsely modeled after biological foveas: There are comparatively many comparatively small receptive fields near the center of the fovea. There are comparatively few comparatively large receptive fields in the periphery of the fovea. See figure 1.

Multiple resolutions offer a potential for applying maximal resolution to each part of the pixel plane by simply moving the fovea center there. The low-resolution parts of the fovea are useful for detecting coarse structure in the visual scene. The high-resolution parts are useful for detecting details. Later on we will show that the low-resolution parts may trigger system actions which lead the high-resolution parts of the fovea to potentially relevant parts of the plane. These actions may be viewed as *attention guiding* actions.

At each time step of a multi-step target detection process from each receptive field we extract one input value for our learning system. Such an input value is simply the average value of the pixels currently covered by the corresponding field (there is no sophisticated pre-processing).

The goal is adaptive target detection. In our case the targets are pre-defined details of 2-dimensional objects which can be arbitrarily translated and/or rotated on the pixel plane. In the beginning of each target detection task the fovea is placed on a randomly chosen part of the pixel plane. Then there is a limited number of time steps during which the system can generate a finite sequence of fovea movements. At the end of the target detection process the fovea should be placed directly above the target (showing the correct rotation). The final position and rotation of the fovea represent the desired information. During training, the only goal-specific information is given by predefined *desired* input values which correspond to those input values obtained by placing the fovea directly on the target. The final input values should match the desired input values. No informed teacher provides knowledge about useful fovea movements, making the task is a 'reward-only-at-goal' task.

The first network of our learning system is called the controller C. The input values provided by the retina are the inputs of C. At each time step of a multi-step target detection process C produces outputs (actions) which serve to control movements of the artificial retina on the pixel plane. In general, a movement causes new input values. This kind of feedback is called *external feedback*. See figure 2.

The problem is to find a mapping from retina inputs to control actions making the system find the target at the end of each finite target detection process. Note that we are not looking for a system that finds the shortest path from the current position of the retina to the target. In fact, with many practical problems the current fovea input will *not* provide enough information for determining the direction to the target. In such cases we want the system to

learn to generate moves causing new inputs that allow to *continue with 'more informed' moves*. This can be interpreted as active perception and attention shifting.

Our desired mapping has to be implemented by C. Note that C cannot be trained by simple supervised learning. Simple supervised learning would require an external teacher providing the desired output actions at each time step. In our case, however, the only external information is about 'how the target looks'. There only is one final desired *input*. (Control theory calls this a 'terminal control problem'.)

From the difference between the desired and the actual *input* at the end of a trajectory we somehow would like to generate gradient information for the *output* units of the controller. This would require to somehow propagate errors from the input units back 'through the environment', which is not possible. The remedy is to consider C's output units as *hidden* units of a larger dynamic recurrent network obtained by the following procedure:

1. Introduce an additional *model network* M (with a separate learning procedure) for 'bridging the gap' between the outputs of the controller and its inputs at the next time step: M is trained to *emulate* the visible environmental dynamics by predicting the next input, given the current controller input and output. See figure 2. M serves to approximate the environmental dynamics by a differentiable mapping which will be used for the temporal credit assignment process of C. (We will see that M need not be a perfect predictor to allow C to discover perfect solutions.) *No informed teacher is required for* M: M can be trained by generating random fovea movements and observing the consequences.

2. Now identify C's output units with the corresponding input units of M and identify M's ouput units with the corresponding input units of C. See figure 3. Gradient information for the weights of the control network now can be back-propagated from M's final prediction through the model network down into the control network and back through the model network etc. according to the 'unfolding in time' algorithm [18] [9]. An important difference to conventional 'back-propagation through time' (with a single recurrent network) is that the weights of the model network remain fixed during this procedure.

In different contexts and with different degrees of generality the basic principle for credit assignment by *system realization* and *'gradient descent through a frozen model network'* has been previously described by Werbos [19], Jordan [2], Munro [5], Robinson & Fallside [8], Nguyen & Widrow [6], and Schmidhuber [10] [11] [15].

The only work by other authors that also addresses the problem of *learning* active perception in reactive environments (and that we are aware of) is the work of Whitehead and Ballard [20]. Their system uses adaptive actions that can bind 'markers' to certain features of an environmental state. Markers dynamically mask or emphasize inputs from the visible environment. This is analogous to our fovea-guiding actions which dynamically change the input such that certain environmental details become visible, while others disappear. With Whitehead

and Ballard's system the learning of active perception is based on an adaptive control technique for delayed reinforcement learning called 'Q-learning' [16].

Our approach implements an adaptive control technique for 'reward-only-at-goal' tasks which is quite different from those reinforcement learning control architectures used by Whitehead and Ballard. Our approach is gradient-based. It tries to provide an error gradient for the controller outputs by propagating the final input error through time-varying instances of a differentiable approximation of the environment (namely the model network).

If objects in a visual scene may occupy random positions then it will be impossible for the model network to predict exactly the future fovea inputs from previous ones. Unlike with e.g. the 'truck backer upper' [6] both C and M never 'see' the complete state of the environment, but only some local details. But this is what active perception is good for: The focus of attention should be shifted to parts of the scene allowing to acquire more detailed (domain-dependent) information about how to go on in the target detection process. The main task of the model network is to help the controller to move the fovea into regions of the plane which *allow to continue with more informed moves.* (Although one can not exactly predict what one will see after moving one's eyes to the door, one is setting the stage for additional eye-movements that help to recognize an entering person.) If the things one attends to never provided *unpredicted* information, the concept of attention would make no sense. One might say that if the situation was such that the model network could be trained to always make perfect predictions, *there would be no need for a model network.* In that case a *single* network would be able to store *all* information about the environment. Thus for all interesting cases the model network *necessarily* has to remain imperfect.

So unlike with the 'truck backer upper' problem [6] it is not intended to make M a perfect predictor whose output could replace the input from the environment (in that case not much would be gained compared to the static approach to target detection). But, an imperfect model network still can capture enough of the environmental dynamics to allow the controller to learn perfect solutions, as will be seen in the experimental section. The reason is: *It suffices if the inner products of the approximated gradients (based on an inaccurate model) for C and the true gradients (according to a hypothetical perfect model) are always positive* (see also [2]). Even if these inner products are not always positive but only 'in most cases', performance improvement can be expected.

2.2 Formal Details

In the comparatively simple case considered here, the controller C is a standard back-propagation network. There are discrete time steps. Each fovea trajectory involves k discrete time steps 1 ... k. At time step t of trajectory p, C's input is the real-valued vector $x_p(t)$ which is determined by sensory perceptions from the artificial 'fovea'. C's output at time step t of trajectory p is the vector

$c_p(t)$. At each time step t motoric actions like 'move fovea left', 'rotate fovea' are based on $c_p(t)$. The actions cause a new input $x_p(t+1)$. The final desired input d_{pfin} of the trajectory p is a predefined activation pattern corresponding to the target to be found in a static visual scene. The task is to sequentially generate fovea trajectories such that for each trajectory p d_{pfin} matches $x_p(k)$. The *final input error* e_{pfin} at the end of trajectory p (externally interrupted at time step k) is

$$e_{pfin} = (d_{pfin} - x_p(k))^T (d_{pfin} - x_p(k)).$$

Thus e_{pfin} is determined by the *differences between the desired final inputs and the actual final inputs.*

In order to allow credit assignment to past output actions of the control network, we first train the model network M (another standard back-propagation network) to emulate the visible environmental dynamics. This is done by training M at a given time to predict C's next input, given the previous input and output of C. The following discussion refers to the case where both M and C learn in parallel. In some of the experiments below we use two separate training phases for M and C. However, the modifications are straight-forward and mainly notational.

M's input vector at time t of trajectory p is the concatenation of $c_p(t)$ and $x_p(t)$. M's real-valued output vector at time t of trajectory p is $m_p(t)$, where $\mid m_p(t) \mid = \mid x_p(t) \mid$. (Here $\mid x \mid$ is the dimension of x, M has as many output units as there are input units for C.) $m_p(t)$ is M's prediction of $x_p(t+1)$. The error of M's prediction at time $0 < t < k$ of trajectory p is

$$E_p(t) = (x_p(t+1) - m_p(t))^T (x_p(t+1) - m_p(t)).$$

M's goal is to minimize $\sum_{p,t} E_p(t)$, which is done by conventional back-propagation [17][7][4][9]:

$$\triangle W_M^T = -\alpha_M \frac{\partial \sum_{p,t} E_p(t)}{\partial W_M}.$$

Here W_M is M's weight vector, $\triangle W_M$ its change caused by the back-propagation procedure, and α_M is M's constant learning rate. (In the experiments described below we will deviate from pure gradient descent by changing M's weights after each time step of each trajectory.)

C's training phase is more complex than M's. It is assumed that $\sum_p e_{pfin}$ is a differentiable function of W_C, where W_C is C's weight vector. To approximate

$$\frac{\partial \sum_p e_{pfin}}{\partial W_C},$$

it is assumed *that M with fixed W_M can substitute the environmental dynamics.* As described below, M is used to approximate the desired partial derivative, but only C's weights are allowed to change, W_M remains fixed. The 'unfolding

in time' algorithm [9][18] is applied to the recurrent combination of M and C (figure 3) to compute

$$\triangle W_C^T = -\alpha_C \sum_p \left(\frac{\partial m_p(k)}{\partial W_C} \right)^T (d_{pfin} - x_p(k)).$$

Here $\triangle W_C$ is W_C's increment caused by the back-propagation procedure, and α_C is the learning rate of the controller. Note that the differences between target inputs and *actual* final inputs at the end of each trajectory are used for computing error signals for the controller. We do *not* use the differences between desired final inputs and *predicted* final inputs.

To apply the 'unfolding in time' algorithm [9][18] to the recurrent combination of M and C, do the following:

For all trajectories p:

1. During the activation spreading phase of p, for each time step t of p create a copy of C (called $C(t)$) and a copy of M (called $M(t)$).

2. Construct a large 'unfolded' feed-forward back-propagation network consisting of $2k$ sub-modules by doing the following:

2.a) For $t > 1$ replace each input unit u of $C(t)$ by the unit in $M(t-1)$ which predicted u's activation.

2.b) For $t \geq 1$: Replace each input unit of $M(t)$ whose activation was provided by an output unit u of $C(t)$ by u.

3. Propagate the difference $(d_{pfin} - x_p(k))$ back through the entire 'unfolded' network constructed in step 2. Change each weight of C in proportion to the sum of the partial derivatives computed for the corresponding k connection copies in the unfolded network. Do not change the weights of M.

Since the weights remain constant during the activation spreading phase of one trajectory, the practical algorithm used in the experiments does not really create copies of the weights. It is more efficient to introduce one additional variable for each controller weight: This variable is used for accumulating the corresponding sum of weight changes. During trajectory execution, it is convenient to push the time-varying activations of the units in M and C on stacks of activations, one for each unit. During the back-propagation phase these activations can be successively popped off for the computation of error signals.

2.3 Dynamic Equilibria Through the Environment

Since the task is to stop the fovea as soon as a certain detail of the environment is focussed, one can draw an interesting analogy to *static* equilibrium networks

(like e.g. the Hopfield network, or the Boltzmann machine). To see this, consider the whole combined system consisting of retina, controller, and pixel plane: A given weight vector for C together with a given visual scene defines an 'energy landscape' where the attractors should correspond to solutions for the target detection task.

The main difference to conventional equilibrium networks is the fact that the dynamic equilibrium corresponding to a certain attractor involves *external feedback*. A mathematical analysis of such energy landscapes seems to be difficult, since it has to take domain-dependent details of the environment into account.

3 EXPERIMENTS

3.1 Target Detection Without Rotations

Consider again figure 1. A visual scene was made of a dark object on a white background which was placed on a 512 x 512 pixel field. Instead of using hundredthousands of input units (as in a straight-forward inefficient static approach) only 40 input units for C were employed. These were sitting on the fovea (a two-dimensional artificial retina) which was controlled by the activations of four output units of the control network: There was one output unit for each of the directions 'up', 'down', 'left', and 'right'. At each time step the activation of each output node was mapped (by a multiplication operation) to the interval between 0 pixels and 20 pixels. The result was interpreted as the length of a vector pointing in the corresponding direction. A move was computed by adding the four vectors. The fovea diameter was about equal to the object size. Figure 1 shows a typical visual scene and the *receptive fields* of the 40 input units. At a given time step the activation of an input unit was computed by simply averaging the values of the pixels (black = 1, white = 0) covered by its receptive field. All non-input units in the system employed the logistic activation function $f(x) = \frac{1}{1+e^{-x}}$. M had a layer of $40 + 4 = 44$ input units, a layer of 40 hidden units, and a layer of 40 output units. C had a layer of 20 hidden units. Both C and M were *fully* forward-connected. Unlike with the more complicated situations described in [8][12][10][11] we did not allow internal feedback *within* C or M. In the beginning all weights were randomly initialized between -0.1 and 0.1. Both α_C and α_M were set equal to 0.1.

With this experiment, there were two separate training phases for M and C. First M was trained: For 50000 training cycles the fovea was randomly placed in the environment of the object, and a move was generated according to a uniform distribution of possible controller outputs. As mentioned above, we deviated from 'real' gradient descent by changing M's weights after each training cycle. After the training phase M's average error was about 10 percent. Now M's weights were fixed and C's training phase (involving 20000 'trials') began.

In the beginning of each 'trial' the object occupied a randomly chosen po-

sition in the pixel field. Again the fovea was randomly placed near the object such that the latter was partially overlapped by some of the receptive fields of the input units (figure 1). Then C generated a fovea trajectory. Whenever the fovea left the pixel plane its receptive fields received 'white' zero input. C's final input error was determined as described above, and the 'unfolding in time' algorithm was applied. During training k was set equal to 5 (this corresponds to $5 * (2 + 2) = 20$ 'layers' in the 'unfolded' network). After training, 50 time steps per trajectory were allowed.

The system described above was able to learn (without a teacher) correct sequences of fovea movements although the model network often made erroneous predictions. (The precondition for a successful trajectory was a partial overlap between the area covered by the object and the area covered by the 'retina' in the beginning of some trajectory.) At the end of a successful trajectory the fovea used to have moved towards the target part of the object. (In figure 4 the fovea center at some time step is given by the center of some arrow.) The accuracy was nearly perfect: In most cases the difference between the desired position and the actual position was not greater than one or two pixels.

Note that the fovea typically did *not* find the *shortest* path to the target. It could not, because the it saw just a part of the scene and usually did not receive enough information to determine the direction to the target. *Instead it often developed a preference for edges.* This is presumably due to the fact that with many of our training objects it is a good strategy to follow the outer boundary line until a new visual cue comes into sight.

Each of the 50-steps trajectories depicted in the figures took about one second real time on a *SUN SPARC station* (including graphics output). Using a fully parallel approach for solving similar target detection problems (by considering all pixels at one single time step) would require orders of magnitude more execution time (and probably much more training cycles for solving the problem of translation invariance, however, due to limited computer time we were not able to test this experimentally).

3.2 One Network for Various Targets

By providing an additional constant controller input which remains time invariant during the generation of some fovea trajectory, various targets can be specified for various trajectories.

The number of C's input units was doubled: For each original input unit there was another input unit whose constant activation defined the desired activation at the end of a fovea trajectory (the goal). (This goal-defining feature is also relevant for 'higher-level' sub-goal generating processes to be addressed later.) M remained unchanged, the same parameters as above were used for the training phase.

The controller was able to learn to look for parts of a scene which matched the time invariant input. See figure 5 for an illustration of trajectories leading

to different targets in the same scene.

3.3 Target Detection Including Rotations

Two additional output units for C were introduced for controlling fovea *rotations* (around the fovea center), one for each of the directions 'clockwise' and 'counter-clockwise'. Thus the number of M's input units increased to 46. At a given time, a clockwise rotation was computed by mapping (through a multiplication operation) the current activation of the first additional output unit to a rotation angle between 0 and 50 degrees. The counter-clockwise rotation was computed by mapping the current activation of the second additional output unit to a rotation angle between -50 and 0 degrees. The final rotation was the sum of both rotations. The same initialization conditions and learning rates as with the translation experiments were employed. As it was expected, the learning of fovea trajectories which include rotations proved to be more difficult than the learning of pure translation sequences. 100000 training examples for M and 20000 training trajectories for C were employed.

Consider figures 6-9: In the beginning of some trajectory both the fovea and the test object (a triangle) were arbitrarily positioned and rotated in the pixel field. (However, the receptive fields of the input units partially overlapped the object.) The fovea rotation at each time step of some trajectory is indicated by the direction of an arrow. The task was to generate a fovea trajectory which lead the center of the fovea to a predefined point near the center of the triangle such that the arrow pointed towards the corner with the smallest angle.

The experiments show that the learning of successful fovea trajectories involving translations and rotations is possible, although M ususally makes erroneous predictions. See [1] for additional experiments.

It should be noted that we currently cannot answer general questions like: How many input units and how many hidden units are necessary for which kind of visual scenes? What are the optimal learning rates?

3.4 Target Tracking

Further experiments with the same objects as above showed that the system is well-suited for target tracking. The desired detail of the moving object soon is focussed and tracked, as long as the objects velocity does not excess the maximal fovea velocity. Note that this is just a by-product of the learning procedure, there is no need for additional training.

3.5 Parallel Learning of C and M: The Need for Probabilistic Output Units

With the experiments reported in the last sections there were separate training phases for M and C. The search element that usually is incorporated within

reinforcement learning systems by using probabilistic activation rules was buried in the random search of the first phase.

For realistic large scale applications it is highly desireable that M and C learn in parallel. In general the model network will not be able to explore *all* possible combinations of inputs and actions and their consequences. The control network should already start learning with an incomplete representation of the external dynamics in the model network. M should concentrate on those parts of the external dynamics that are necessary for achieving C's goals. Just like Kohonen's self organizing feature maps [3] dedicate more storage capacity for fine grained representation of common similar inputs, M should dedicate more storage capacity and time for fine grained modeling of those aspects of the world that are likely to be relevant for the system's main goal. (See [11] for more reasons for parallel on-line learning of M and C.)

We conducted some experiments with on-line learning. It was found that two interacting conventional *deterministic* networks in the style of [2] and [6] were *not* appropriate. Usually a deterministic system soon became trapped in a state where the controller never shifted the fovea towards regions which allowed the model network to collect new relevant information about the external world. This is called the *deadlock* problem.

To attack the *deadlock* problem, we introduced some modifications for the controller, in order to provide it with *explicit* search capabilities. Each of the output units was replaced by a little network consisting of two units, one giving the mean and the other one giving the variance for a random number generator which produced random numbers according to a continuous distribution. (We approximated a Gauss distribution by a Bernoulli distribution.) Weight gradients were computed by applying William's concept of 'back-propagation through random number generators' [21].

It was found that within 100000 trials such an on-line learning system was able to learn appropriate fovea trajectories (like e.g. in figure 4). As it was expected, after training the model network was a good predictor *only* for those situations which the controller typically was confronted with.

With these experiments, the on-line approach did not significantly improve efficiency. So the main contribution of this section is the demonstration that the introduction of probabilistic output units can make on-line learning possible.

4 ONGOING AND FUTURE RESEARCH

The approach described above certainly does not solve all problems of adaptive attentive vision. So far our system has been tested only with fairly simple visual scenes involving objects with a rather simple geometrical shape. It is not clear how well the system will do with more complicated scenes. However, there are some promising directions for future research.

4.1 Scenes With Multiple Objects

Scenes with multiple objects or objects with rich internal details require either recurrent connections in both M and C or some other mechanism for escaping certain cases of *local minima*. Local minima can be caused by parts of the pixel plane that look similar to the target input, while the nearby environment does not. In such cases the *relevant* external feedback through the environment becomes *non-Markovian*. For such situations, additional experiments (not reported here) with a recurrent M and a recurrent C were conducted. It turned out that internal feedback within M and C sometimes can lead to success in cases where the simple approach fails [1]. However, there is an approach which in the long run might prove to be even more promising: The adaptive on-line generation of appropriate sub-goals. Some first work in this direction already has been done [13]. By using the above-mentioned concept of goal-defining input units with time-invariant activations, we intend to apply *adaptive sub-goal generators* to the problems of 'local minima' that can arise during the target detection process.

4.2 Methods of Temporal Invariances.

To smoothen the error surface of an attentive vision system as described above, one can *impose temporal smoothness constraints on the input units*. This can be done by constructing a new error function by *adding differences in successive fovea inputs* to the final input error observed at the end of a fovea trajectory. (The approach is reminiscent of Jordan's work [2], however, Jordan imposes temporal constraints on the *output* units.)

The effect is that the system develops a preference for temporal invariances in input space. For attentive vision, such temporal invariances can be caused e.g. by fovea movements that follow edges. Thus an unsupervised element (a search for regularities) is introduced into the learning process. (Trivial temporal invariances obtained by stopping the fovea are excluded by the goal directed part of the complete error function.)

An empirical motivation for introducing an explicit preference for temporal invariances is given by the experimentally observed fact that even without such a predefined preference the system liked to generate fovea trajectories following edges.

4.3 Implications for Learning Selective Attention in the General Case: An Outlook

The system described above (which learns by using the principle of system realization) as well as Whitehead and Ballard's system (mentioned in section 1) can be viewed as implementing selective attention by some sort of external feedback. The system described in [14], which implements *'curiosity'* and *'boredom'*

by means of adaptive dynamic attention depending on the amount of a model network's ignorance about the external dynamics, also is based on external feedback.

A generalization of the method described above would work as follows. *For each input unit of some controller introduce an output unit that gates the current activation of the corresponding input unit at every time step. Train a model network to predict the context dependent effects of suppressing certain input units and emphasizing others. Use system realization as above for* learning *dynamic selective attention to those input units that are* relevant *in the context of the current goal.*

5 CONCLUSIONS

Previous approaches to pattern recognition with neural networks emphasized the parallel 'static' aspects of information processing. However, even in apparently static domains as target detection in stationary environments much can be gained by introducing sequential elements and dynamic selective attention.

This paper demonstrates that the principle of system realization and gradient descent through a model network can be used for learning certain cases of dynamic selective attention. The context is given by attentive vision: It is demonstrated that an *imperfect* model network which emulates the fovea dynamics can contribute for learning *perfect* solutions to certain target detection problems. However, the concept of system realization is general enough to allow less specialized approaches to selective attention than the one presented in this paper.

References

[1] R. Huber. Selektive visuelle Aufmerksamkeit: Untersuchungen zum Erlernen von Fokustrajektorien durch neuronale Netze, 1990. Diplomarbeit, Institut für Informatik, Technische Universität München.

[2] M. I. Jordan. Supervised learning and systems with excess degrees of freedom. Technical Report COINS TR 88-27, Massachusetts Institute of Technology, 1988.

[3] T. Kohonen. *Self-Organization and Associative Memory.* Springer, second edition, 1988.

[4] Y. LeCun. Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris*, pages 599–604, 1985.

[5] P. W. Munro. A dual back-propagation scheme for scalar reinforcement learning. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society, Seattle, WA*, pages 165–176, 1987.

[6] Nguyen and B. Widrow. The truck backer-upper: An example of self learning in neural networks. In *IEEE/INNS International Joint Conference on Neural Networks, Washington, D.C.*, volume 1, pages 357–364, 1989.

[7] D. B. Parker. Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT, 1985.

[8] T. Robinson and F. Fallside. Dynamic reinforcement driven error propagation networks with application to game playing. In *Proceedings of the 11th Conference of the Cognitive Science Society, Ann Arbor*, pages 836–843, 1989.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, 1986.

[10] J. H. Schmidhuber. Learning algorithms for networks with internal and external feedback. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Proc. of the 1990 Connectionist Models Summer School*, pages 52–61. San Mateo, CA: Morgan Kaufmann, 1990.

[11] J. H. Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90 (revised), Institut für Informatik, Technische Universität München, November 1990. (Revised and extended version of an earlier report from February.).

[12] J. H. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 253–258, 1990.

[13] J. H. Schmidhuber. Towards compositional learning with dynamic neural networks. Technical Report FKI-129-90, Institut für Informatik, Technische Universität München, 1990.

[14] J. H. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In J. A. Meyer and S. W. Wilson, editors, *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227. MIT Press/Bradford Books, 1991.

[15] J. H. Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 500–506. San Mateo, CA: Morgan Kaufmann, 1991.

[16] C. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, 1989.

[17] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[18] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1, 1988.

[19] P. J. Werbos. Backpropagation and neurocontrol: A review and prospectus. In *IEEE/INNS International Joint Conference on Neural Networks, Washington, D.C.*, volume 1, pages 209–216, 1989.

[20] S.D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. Technical Report 331, University of Rochester, Dept. of Comp. Sci., 1990.

[21] R. J. Williams. On the use of backpropagation in associative reinforcement learning. In *IEEE International Conference on Neural Networks, San Diego*, volume 2, pages 263–270, 1988.

LIST OF FIGURE CAPTIONS

Figure 1. A typical visual scene. The diameters of the receptive fields of the retina's input units are indicated by circles.

Figure 2. An artificial fovea provides inputs for a control network which is able to move the fovea around. A model network is trained to predict the next input from the current input and the current controller action.

Figure 3. By 'substituting the model network for the environment' we obtain a recurrent combination of control network and model network. This new recurrent network is used for computing controller gradients by means of the 'unfolding in time' algorithm.

Figure 4. Translations: Examples of fovea trajectories leading from various start positions to the target, which is the center of the crossing point in the letter '4'. No teacher told the fovea how to do that! Typically the system did not find the shortest path to the target. It developed a preference for edges.

Figure 5. One controller for various targets specified by an additional constant input: Examples of fovea trajectories leading from various start positions to different targets. The first target is near the left corner of the triangle. The second target is near the lower corner.

Figure 6. A triangle which may be arbitrarily rotated and translated in the pixel plane. The triangle is partly covered by some of the receptive fields of the moving fovea.

Figures 7 and 8. Examples of fovea trajectories leading from the outside of the object to the target. (For clarity, the fovea positions are indicated only for a fraction of all time steps.) No teacher told the system how to do that!

Figure 9. The fovea pushing backwards to the target on a noisy pixel plane.