Neural Networks 22 (2009) 586–592

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2009 Special Issue

Goal-directed learning of features and forward models

Sohrab Saeb *, Cornelius Weber, Jochen Triesch

Frankfurt Institute for Advanced Studies (FIAS), Goethe University, Frankfurt am Main, Germany

ARTICLE INFO

Article history:
Received 6 May 2009
Received in revised form 2 June 2009
Accepted 25 June 2009

Keywords:
Reinforcement learning
Forward model
Feature learning
Recurrent neural network

ABSTRACT

The brain is able to perform actions based on an adequate internal representation of the world, where task-irrelevant features are ignored and incomplete sensory data are estimated. Traditionally, it is assumed that such abstract state representations are obtained purely from the statistics of sensory input for example by unsupervised learning methods. However, more recent findings suggest an influence of the dopaminergic system, which can be modeled by a reinforcement learning approach. Standard reinforcement learning algorithms act on a single layer network connecting the state space to the action space. Here, we involve in a feature detection stage and a memory layer, which together, construct the state space for a learning agent. The memory layer consists of the state activation at the previous time step as well as the previously chosen action. We present a temporal difference based learning rule for training the weights from these additional inputs to the state layer. As a result, the performance of the network is maintained both, in the presence of task-irrelevant features, and at randomly occurring time steps during which the input is invisible. Interestingly, a goal-directed forward model emerges from the memory weights, which only covers the state–action pairs that are relevant to the task. The model presents a link between reinforcement learning, feature detection and forward models and may help to explain how reward systems recruit cortical circuits for goal-directed feature detection and prediction.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

Internal representation, or the representation of the outside world inside our brain, is the substrate based on which we perform actions. This representation is formed in such a way that it allows reliable functionality under various environmental situations. Our brain is able to provide this adequate representation by *ignoring task-irrelevant* and *completing incomplete* sensory information that is relevant to the goal.

Mammalians have subcortical structures, known as neuromodulatory systems, which regulate diverse populations of neurons by means of different classes of neurotransmitters (Hasselmo, Wyble, & Fransen, 2003). These structures are able to alter responses and internal states of the organism to induce a goal-directed behavior (Jog, Kubota, Connolly, Hillegaart, and Graybiel (1999) and Hyman, Malenka, and Nestler (2006); for a review from a roboticist's perspective see Krichmar (2008)). Reinforcement learning (RL) models (Sutton & Barto, 1998) are consistent with biological findings about the function of the dopaminergic neuromodulatory system, in predicting future rewards and punishments (Montague, Dayan, & Sejnowski, 1996; Montague, Hyman, & Cohen, 2004; Schultz, Dayan, & Montague, 1997). By incorporating a temporal difference error (delta) signal, which represents the activity of dopaminergic neurons, these models are able to discover by trial and error how an organism should behave in order to get the most reward and the least punishment (Sutton & Barto, 1998). However, Redgrave and Gurney (2006) point out that dopamine neurons also increase their activity in the presence of novel, but not necessarily reward predicting stimuli. Laurent (2008) addressed this issue in a simple model to reconcile these findings within a reinforcement learning framework. Therefore, reinforcement learning seems an appropriate framework for both neuroscientists dealing with reward-timing dependent behavioral tasks and engineers developing intelligent systems that adapt to tasks.

It has been proposed that unsupervised learning, e.g. Hebbian plasticity, mostly happens in the cortex (Rauschecker & Singer, 1979), while reinforcement learning in the basal ganglia (Schultz, 1998). Accordingly, the cortex extracts features to yield a representation that is suitable for, but independent of, reinforcement learning by the basal ganglia (Doya, 1999). However, more recent biological studies suggest that reinforcement learning is not limited to basal ganglia, but it also affects cortical areas, which implies a dependence of this representation on the task (Kennerley & Wallis, 2009; Schoups, Vogels, Qian, & Orban, 2001; Shuler & Bear, 2006). In a computational study, we showed that the effect of reinforcement learning on feature detection gives rise to goal-directed feature detectors (Weber & Triesch, 2009). We used a combination of the delta signal and Hebbian-like plasticity to train the feature detector parameters, in addition to utilizing this signal for adapting

* Corresponding address: Raum 2.204, Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany. Fax: +49 69 798 47611.
E-mail address: saeb@fias.uni-frankfurt.de (S. Saeb).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.06.049

the actor network weights. As a result, the feature detection network learned to disregard *task-irrelevant* information and attend to those feature(s) which lead the agent to reward.[1]

Having found a possible way of dealing with *task-irrelevant* information, now the question is, how does the brain solve the problem of *incomplete* sensory information? In artificial intelligence, the case of incomplete sensory data leads to the perceptual aliasing problem. An essential prerequisite for a reinforcement learning agent to learn an optimal policy is that the agent–environment interaction be a Markov decision process (MDP). This implies that the agent's future state is predictable only based on the current state and chosen action. When perceptual aliasing occurs, the agent–environment interaction is no longer an MDP, but a partially-observable MDP (POMDP) (Smallwoo & Sondik, 1973). Different solutions such as incorporation of recurrent architectures (Bakker, 2002; Whitehead & Lin, 1995), belief networks (Jog, 1995), and state space filters (Nagayoshi, Murao, & Tamaki, 2006) have been proposed in this regard, aiming at building a consistent state representation which can be modeled as an MDP.

What is common in all of these solutions is the exploitation of previous state representations in estimating the current state. Does the brain have necessary substructures to provide such a solution? Theoretically, a working memory holds previous state representations based on which a currently missing representation can be predicted. The circuits that provide such prediction are also called internal forward models. There are pieces of evidence for the existence of such circuits in various brain regions (for a model review, see Downing, 2009) such as lateral prefrontal cortex (Mushiake, Saito, Sakamoto, Itoyama, & Tanji, 2006). These brain areas, on the other side, are among those areas which are influenced by the activity of dopaminergic neurons (Hasselmo et al., 2003). Therefore, it is an interesting question to ask if these forward models can emerge as a result of dopaminergic signals.

In this paper, we will show this possibility in a computational framework. To this end, we consider a scenario that requires the prediction of missing sensory data, as well as ignoring irrelevant features. We extend the goal-directed feature learning network (Weber & Triesch, 2009) by involving a memory layer (Fig. 1), and train all weight parameters by a delta signal. We use a simple and an extended architecture based on recurrent neural networks. The simple architecture uses only the previous-step state representation, similar to the Elman network (Elman, 1990). The extended architecture uses the previously performed actions in addition to the previous states, by introducing Sigma-Pi weights. We will show that a goal-directed forward model emerges in our recurrent networks.

2. Model architecture

The model consists of an input layer holding a sensory vector $\mathbf{I}^t$ at time step t. A hidden feature layer encodes a very sparse state vector $\mathbf{s}^t$, and an output action-selection layer with activation vector $\mathbf{a}^t$ represents the selected action at t by one activated unit. For the extended architecture (Fig. 1-a), two memory layers hold the state and the action vectors of the previous time step, $\mathbf{s}^{t-1}$ and $\mathbf{a}^{t-1}$, respectively, while the simple architecture only uses the previous state vector (Fig. 1-b). The model has full connectivity between adjacent layers.

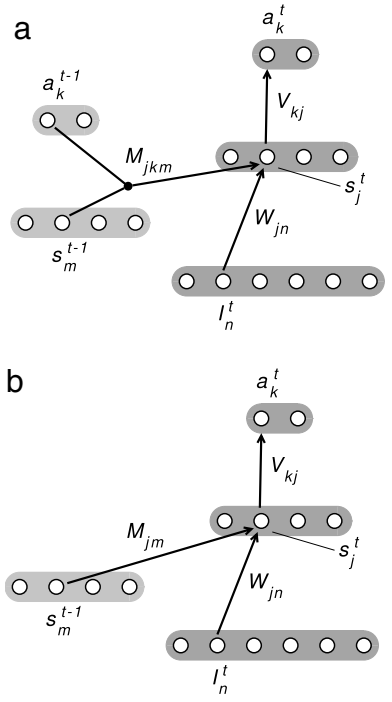

Fig. 1. Model structure drawn schematically for (a) extended architecture; (b) simple architecture. Between connected layers, there is a full connectivity, of which only one example connection is shown.

For the extended architecture, the input to the hidden layer consists of weighted connections **W** from the input, and Sigma-Pi weighted connections **M** from the previous state and action vectors, and is computed as:

$$h_j^t = \sum_n W_{jn} I_n^t + \sum_{k,m} M_{jkm}\, a_k^{t-1}\, s_m^{t-1}. \tag{1}$$

For the simple architecture, this input is computed as:

$$h_j^t = \sum_n W_{jn} I_n^t + \sum_m M_{jm}\, s_m^{t-1}. \tag{2}$$

The activation of the hidden layer neurons for both architectures is obtained using a softmax function:

$$s_j^t = \frac{\exp(\beta h_j^t)}{\sum_p \exp(\beta h_p^t)} \tag{3}$$

where the inverse temperature β controls the amount of exploration versus exploitation. Since we are interested in having a winner-take-all (WTA) behavior in this layer, we choose a sufficiently large value for β (we will use $\beta = 100$ in our simulations) so that the softmax function approaches a WTA function.

The input to the action layer is computed using weighted connections from the state vector:

$$g_k^t = \sum_j V_{kj}\, s_j^t. \tag{4}$$

Finally, an action a_k is chosen based on the probability provided by another softmax function:

$$\mathrm{P}(a_k^t = 1) = \frac{\exp(\beta' g_k^t)}{\sum_p \exp(\beta' g_p^t)}. \tag{5}$$

At the beginning of training, we set the inverse temperature $\beta' = 10$ for more exploration in the state space. After 10,000 trials, we set $\beta' = 200$ to exploit the knowledge acquired during

[1] It is worth noting that reward related action selection structures in the brain are furthermore organized into limbic, associative and motor compartments, with distinct dopamine cells (Haber, Fudge, & McFarland, 2000). This may allow to combine actions in parallel and hierarchically. Such complexity is outside the scope of our model, which implements only a single global dopamine signal. For a deeper biological insight into action formation and its associated questions, the reader is referred to Graybiel (2008) and Redgrave and Gurney (2006).

the exploration phase, and consequently increase the agent's performance.

3. Temporal difference learning

The goal of reinforcement learning is to find an optimal map $\Pi : \mathbf{S} \mapsto \mathbf{A}$ called policy, from a set of states $\mathbf{S}$ to a set of actions $\mathbf{A}$, such that to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead it must discover which actions yield the most reward by trial and error. Actions may not only affect the immediate reward, but also induce a transition in state space with the probability of $P_{\mathbf{a}}(\mathbf{s}^{\text{old}}, \mathbf{s}^{\text{new}})$, through that, all subsequent rewards may be obtained. These two characteristics, namely trial-and-error search and delayed reward, are the two most important distinguishing features of reinforcement learning (Sutton & Barto, 1998).

The temporal difference (TD) reinforcement learning approach can be performed by a gradient descent on a dynamic error function, which itself changes with the policy during the course of learning, unlike the static error functions used in supervised learning. This error function is defined as:

$$E^t = \frac{1}{2} \sum_{\mathbf{s}^t, \mathbf{a}^t} P^{\Pi}(\mathbf{s}^t, \mathbf{a}^t)(Q^{\Pi}(\mathbf{s}^t, \mathbf{a}^t) - Q(\mathbf{s}^t, \mathbf{a}^t))^2 \tag{6}$$

where $P^{\Pi}(\mathbf{s}^t, \mathbf{a}^t)$ is the policy-dependent distribution of state–action pairs, $Q(\mathbf{s}^t, \mathbf{a}^t)$ is the current estimate of the value function and $Q^{\Pi}(\mathbf{s}^t, \mathbf{a}^t)$ is the true value function according to the current policy. The value function is estimated at each time step t by the action selection network, as:

$$Q(\mathbf{s}^t, \mathbf{a}^t) = \sum_{j,k} V_{kj}\, a_k^t\, s_j^t. \tag{7}$$

The true value function Q^{Π} is estimated using the current reward signal, r^t, and the future value $Q(\mathbf{s}^{t+1}, \mathbf{a}^{t+1})$ discounted by a factor γ (Sutton & Barto, 1998):

$$Q^{\Pi}(\mathbf{s}^t, \mathbf{a}^t) \simeq r^t + \gamma Q(\mathbf{s}^{t+1}, \mathbf{a}^{t+1}). \tag{8}$$

For simplicity, we define $Q^t = Q(\mathbf{s}^t, \mathbf{a}^t)$ and $Q^{t+1} = Q(\mathbf{s}^{t+1}, \mathbf{a}^{t+1})$. The temporal difference error, δ^t, is calculated as:

$$\delta^t = Q^{\Pi} - Q^t \simeq r^t + \gamma Q^{t+1} - Q^t. \tag{9}$$

To update the network parameters $\mathbf{V}$, $\mathbf{P}$ and $\mathbf{W}$, with the gradient descent approach, we calculate the partial derivatives of the error function with respect to each parameter. In on-line learning, the on-policy distribution $P^{\Pi}(\mathbf{s}^t, \mathbf{a}^t)$ is implicit in the selection of data, therefore there is no need to include it in the gradient calculation. For any parameter Θ the gradient will be in form of:

$$\frac{\partial E^t}{\partial \Theta} = \frac{\partial E^t}{\partial Q^t} \frac{\partial Q^t}{\partial \Theta} = -\delta^t \frac{\partial Q^t}{\partial \Theta}. \tag{10}$$

We start with the upper layer parameters $\mathbf{V}$. Using Eq. (7), we obtain the action weight update as:

$$\Delta V_{kj} \propto -\frac{\partial E^t}{\partial V_{kj}} = \delta^t\, a_k^t\, s_j^t.$$

Next, we calculate the error function derivatives with regard to the lower layer parameters, $\mathbf{W}$ and $\mathbf{M}$. Keeping in mind that W_{jn} influences not only the state neuron s_j but the activations of all state neurons s_k, we have:

$$\Delta W_{jn} \propto -\frac{\partial E^t}{\partial W_{jn}} = -\sum_p \frac{\partial E^t}{\partial s_p^t} \frac{\partial s_p^t}{\partial h_j^t} \frac{\partial h_j^t}{\partial W_{jn}}.$$

Assuming that the action unit i is the activated one, Eq. (7) gives us:

$$\frac{\partial E^t}{\partial s_p^t} = \delta^t\, V_{ip}.$$

For the softmax function of Eq. (3), we use the following identities (Nguyen, unpublished):

$$\frac{\partial s_j}{\partial h_j} = \beta s_j(1 - s_j) \quad \text{and} \quad \frac{\partial s_j}{\partial h_{p,p\neq j}} = -\beta s_p s_j.$$

From Eq. (2), we have:

$$\frac{\partial h_j^t}{\partial W_{jn}} = I_n^t.$$

Together, we obtain the following update rule for the feature weights at time step t:

$$\begin{aligned}\Delta W_{jn} &\propto \delta^t\, \beta\, V_{ij} s_j^t (1 - s_j^t) I_n^t - \delta^t\, \beta \sum_{p,p\neq j} V_{ip} s_p^t\, s_j^t I_n^t \\ &= \delta^t\, \beta\, s_j^t I_n^t \left(V_{ij} - \sum_p V_{ip} s_p^t \right).\end{aligned} \tag{11}$$

For the memory weights of the extended architecture (Fig. 1-a), we have:

$$\frac{\partial h_j^t}{\partial M_{jkm}} = a_k^{t-1}\, s_m^{t-1}$$

and obtain:

$$\begin{aligned}\Delta M_{jkm} &\propto \delta^t\, \beta\, V_{ij}\, s_j^t (1 - s_j^t)\, a_k^{t-1} s_m^{t-1} - \delta^t\, \beta \sum_{p,p\neq j} V_{ip}\, s_p^t\, s_j^t\, a_k^{t-1} s_m^{t-1} \\ &= \delta^t\, \beta\, s_j^t\, a_k^{t-1} s_m^{t-1} \left(V_{ij} - \sum_p V_{ip} s_p^t \right).\end{aligned} \tag{12}$$

For the simple architecture (Fig. 1-b) we have:

$$\frac{\partial h_j^t}{\partial M_{jm}} = s_m^{t-1}$$

and obtain:

$$\begin{aligned}\Delta M_{jm} &\propto \delta^t\, \beta\, V_{ij}\, s_j^t (1 - s_j^t)\, s_m^{t-1} - \delta^t\, \beta \sum_{p,p\neq j} V_{ip}\, s_p^t\, s_j^t\, s_m^{t-1} \\ &= \delta^t\, \beta\, s_j^t\, s_m^{t-1} \left(V_{ij} - \sum_p V_{ip} s_p^t \right).\end{aligned} \tag{13}$$

The first summation term of Eqs. (11)–(13) can be considered as a Hebbian plasticity term modulated with the temporal difference δ^t, except for the factor V_{ij} that arises through backpropagation and that denotes how strong state neuron j contributes to the output.

The second summation term of these equations represents a competitive decay term that has a larger suppressive effect if strong activations s_p in the state layer are paired to large weights V_{ip}. If exactly one feature unit is active, $s_j = 1$, and for all others $s_{p,p\neq j} = 0$, i.e. if a clear winner is found, then the first and the second term cancel, so learning has converged.

Note that this learning rule is not local, first, because of the weight parameter V_{ij} to an action unit and, second, because of the second summation term that sums over all activations in the state layer. A local approximation to Eq. (11) was successfully implemented in our previous study (Weber & Triesch, 2009).

4. Experiments

We designed an experiment that would require a model to develop two properties:

1. It should learn feature detectors to ignore irrelevant sensory input.
2. It should predict the current state based on the previous state and chosen action.

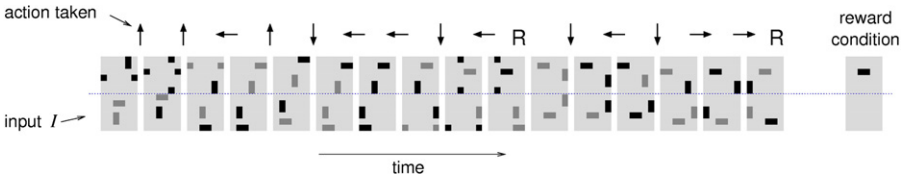

Fig. 2. Data. The sensory input **I** at any time step consists of four features, two horizontal and two vertical bars, one in each half of the input area (separated by the dotted line). Black color indicates visible and gray color invisible bars, which appear as the background to the network. Note periodic boundary conditions. An action moves the features into one of four directions from one time step to the next, as indicated by the arrows (a feature may fail to move with a certain probability). A reward R is given when the "rewarded feature" (here a horizontal bar in the upper half) appears at a specific location (shown as "reward condition"). The network must detect the relevant features (here, the horizontal bars in the upper half), predict its position if invisible, and learn an action strategy. After reward is given, all features are set to a new random position.

At each point in time, four visual features are shown to the model, each being a short bar consisting of two pixels. Two of these features are in the upper half of the visual field, two in the lower half. In each half, one bar is vertically oriented and the other horizontally. Fig. 2 shows example data.

In each trial, the bars are initially located at random positions. When an action is made, all features make a small translation: they move up, right, down or left, depending on the chosen action. There is a 10% probability for each feature to fail doing this translation, assuring that over time the positions of the features with respect to each other are independent. Both halves of the input have periodic boundary conditions, i.e. a feature that leaves one side enters the opposite side. Therefore, the two pixels of a bar may be on opposite sides of each half of the input area.

Of the four bars present at each instant, three are irrelevant. The remaining one is the relevant feature, since its position determines reward: a reward is given whenever the relevant bar is at one specific position, irrespective of the position of the other three bars. This assumption necessitates the presence of a goal-directed feature detector network in the model.

There is a probability of 50% for each bar that it is not seen by the model, i.e. the bars are switched off randomly and independently from each other. This assumption results in a POMDP, which necessitates the second property for the model.

We perform the experiments with three network configurations: First, the extended architecture of Fig. 1-a, which has both state and action memory layers. Second, the simple architecture of Fig. 1-b that has no action memory layer. Third, the feedforward memory-less network presented in Weber and Triesch (2009).

5. Results

The input layer of the network is of size 12×6, consisting of two sub-areas of size 6×6. Hence, given the four stimuli at varying positions, there are $36^4 = 1679\,616$ possible input patterns; more than permissible for direct table based reinforcement learning. The hidden layer consists of 6×6 units; i.e., exactly the number of units required to learn one (the relevant) feature class, which is, all horizontal bars in the upper half of the input. The output layer consists of four action units representing the four directions to move the bars to.

At the beginning, weights were initialized with small random values. The learning algorithm took about 2×10^6 trials to reach a plateau. Fig. 3 shows the network action performance during the training procedure, in terms of the average number of steps taken from the initial random position to the goal. This performance is shown for the extended (Fig. 1-a), simple (Fig. 1-b), and memory-less feedforward networks. The optimal number of steps would be 3 on average if there was no noise. However, with the 10% probability of each action to fail in the experiments, this number amounts to 3.3 steps. While it took on average about 40 steps for a random actor to stumble upon the reward, the learnt extended network reached the goal in around 3.6 steps on average. This number was a bit higher (3.8) for the simple architecture, and noticeably higher (6.4) for the feedforward network.

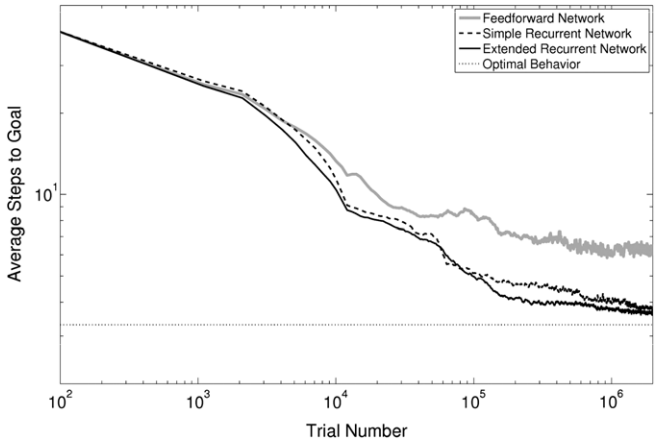

Fig. 3. Action performance for the simple and the extended recurrent architectures, as well as the memory-less feedforward network. The performance at each trial is defined as the number of steps taken by the agent to each of the goal. The values shown here are moving averages over 100-trial-wide windows. The optimal average number of steps (3.3) is shown with dotted horizontal line.

The trained feature weights (for the case of the extended network) are shown in Fig. 4-a. It can be seen that most hidden units have feature weights that match one of the upper horizontal bars. However, relevant features that are far from the rewarded position (i.e. in the corners) are not learnt. Instead, irregularities exist, such as weights to the lower input area, and weights with vertical orientation. Nevertheless, the average energy of receptive field responses to each of the bars (Fig. 4-b) illustrates the state units' preference for the *task-relevant* feature. This energy for each feature is calculated as $\sum_p (\mathbf{W}\mathbf{I}_p) \times (\mathbf{W}\mathbf{I}_p)^{\mathrm{T}}$, where $\mathbf{I}_p$ is a column vector denoting the network input when the feature is at position p of the environment.

To observe the role of recurrent connections, we depicted the networks' input together with its "imagination" of the input, for three successive steps of a trial after learning (Fig. 5). The imagination is calculated by the back-projection of memory output to the network input layer, which is $\mathbf{W}^{\mathrm{T}} \times \mathbf{M} \times \mathbf{S}^{t-1}$. Interestingly, this imagination is close to the real input, which means, whenever the input is absent, the network tries to *predict* it based on its previous internal state and the chosen action. The network makes this prediction only for the *task-relevant* input (upper horizontal bar) and disregards other irrelevant features.

A measurement of the network's prediction accuracy is shown in Fig. 6-a. The prediction accuracy is measured as the inner product between the memory-provided input $\mathbf{M} \times \mathbf{S}^{t-1}$ and the real input image $\mathbf{W} \times \mathbf{I}^t$, when the previous and the current input are both visible. We calculate the prediction accuracies for

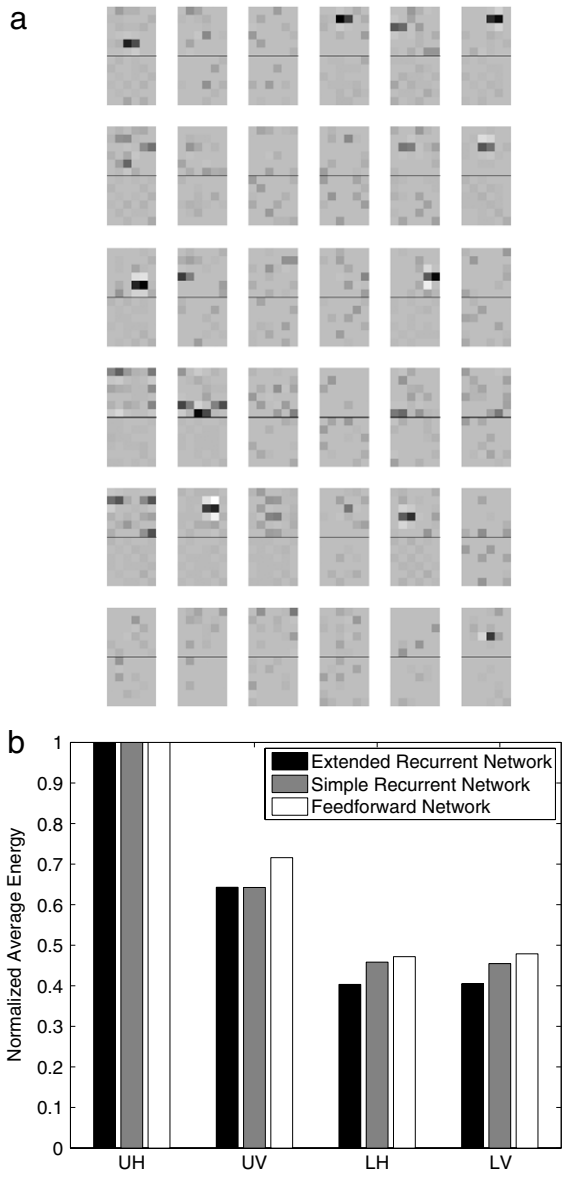

Fig. 4. (a) Receptive fields for the learnt extended architecture. (b) Comparing response energies of different architectures averaged over all possible positions for upper horizontal (UH), upper vertical (UV), lower horizontal (LH) and lower vertical (LV) features. For all of the architectures, the highest energy value is obtained for the relevant feature (UH). This bias is higher for the architectures with memory, implying a role of forward models in developing feature detection circuits.

both the extended and the simple architecture. For the extended architecture, we consider two situations: First, the network follows the learnt policy in a deterministic way, so that the action with the highest probability is chosen. Second, the network follows the "reverse policy", which is defined as the policy of choosing the action with the lowest probability. In all three experiments, the prediction accuracy achieved for the relevant feature is much higher than for irrelevant features. Furthermore, this difference is smaller for the reverse-policy experiment. These two results allow us to conclude that a goal-directed forward model has emerged.

Now we consider prediction accuracies for general Lth-order prediction tasks, where the input has been invisible for L successive time steps prior to the current state. We define the network's preference for predicting the relevant feature as:

$$C(L) = \frac{A_r(L)}{\frac{1}{n(F_{ir})} \sum_{i \in F_{ir}} A_i(L)} \tag{14}$$

where L denotes the prediction order, $A_r(L)$ is the prediction accuracy for the relevant feature, and F_{ir} is the set of irrelevant features. The variation of this preference versus prediction order is shown in Fig. 6-b. For low prediction orders $L = 1, 2$, the simple architecture has higher preference for predicting the relevant feature compared to the extended architecture, but for larger L, the extended network seems to make better predictions of the relevant feature, because it takes into account the actually chosen actions. This explains why the overall performance of the extended architecture is a bit higher than the simple architecture (see Fig. 3).

Putting it differently, after the learning procedure has converged, the agent can rely on a single time step sensory information to estimate the current state. This may lie several time steps in the past, while the input is absent in between. The emerging forward model can act incrementally to estimate the current state in such situations. Fig. 6 shows to what extent the agent can use past information to estimate the current state.

6. Discussion

We extended the goal-directed feature learning model (Weber & Triesch, 2009) to embrace imperfect, noisy sensory observations. While trying to solve the distal reward problem of reinforcement learning, the new model learns both, to *ignore task-irrelevant* and to *predict missing relevant* features of sensory information. It learns a bars task which is challenging even for humans: when a reward is given whenever one of several features appears at a specific position, it is hard to track which feature consistently appears at the same position over several trials. This scenario becomes even more complicated when the bars randomly disappear from the visual field.

The experimental setup used in this study is one step toward getting reinforcement learning to deal with realistic image data with pixel values. Many reinforcement learning approaches still rely on hand-crafted, abstract state spaces to model learning mechanisms of the brain (e.g. see Laurent, 2008; Sprague & Ballard, 2003), whereas in our model the state space is autonomously shaped. Furthermore, our model architecture provides a unique way to interpret the state space activity by back-projecting these activities through feature detection weights and reconstructing the image that is "imagined" by the agent.

The training process may appear excessively long relative to that required for animal learning of goal-directed behavior. One reason for this difference is that our model does not consider any prior knowledge and starts learning from zero. Animal learning may use prior knowledge developed by unsupervised plasticity mechanisms or genetic predispositions to accelerate learning. For instance, the edge-detector property of receptive fields in the primary visual cortex (V1) makes the learning procedure much easier for the animal, while in our model there is no receptive field prior to learning. We have not yet investigated whether efficient exploration strategies, such as the one proposed by Thrun (1992), would lead to faster convergence.

We first implemented an extended recurrent architecture (Fig. 1-a) to allow each action to be considered for predicting the next state. However, we observed that for each position of the relevant bar, almost one specific action was performed according to the learnt policy. Hence, a simple recurrent network (Fig. 1-b) suffices, which can only predict a single consequence of any given state, regardless of the performed action. Nevertheless, in very noisy situations where higher orders of prediction is needed, the feedback from the action space helps the forward model of the

extended recurrent architecture to predict the movement of the task-relevant feature (see Fig. 6-b).

In the classical view of the sensory cortex, neurons such as edge detectors in V1, adapt merely to the statistics of the incoming data. However, this limited view is challenged since reward, such as a few drops of water, can influence neuronal responses in rat V1 (Shuler & Bear, 2006). Even in an experiment without such a reward, if and only if certain stimuli are important for the decision about an action, the corresponding neurons' tuning curves in monkey V1 increase their slope (Schoups et al., 2001). In addition, a recent study reports that information about expected rewards regarding a certain task modulates spatial working memory in ventral lateral prefrontal cortex (Kennerley & Wallis, 2009). Accordingly, neurons in this area maintain task-relevant information across delays, which can be further used by other areas. Such learning effects that have been observed in the cortex suggest an influence of reinforcement learning on the internal representation of sensory information. Our model suggests that such an influence may be responsible for both, making the brain capable of *ignoring task-irrelevant* information, and *predicting task-relevant missing* features.

Traditionally, forward models are trained by supervised learning such that they are able to predict the outcome of performing any action in any state. This approach seems to be highly redundant for an agent that learns a specific task, since in such a situation, the agent does not need to predict the outcome of performing a wrong action with respect to the optimal policy. This task specificity, however, might be considered as a non-desirable effect when multiple tasks with different optimal policies are assigned to the agent. This problem can be resolved by recruiting different task-specific networks for each task. Numerous brain regions, usually in medial and lateral regions of prefrontal cortex, but sometimes in parietal lobes, cerebellum and other subcortical regions, are reported to be involved in task-switching trials (MacDonald, Cohen, Stenger, & Carter, 2000; Monsell, 2003). Merian, Kessler, and Adi-Japha (2008) proposed a task-switching model

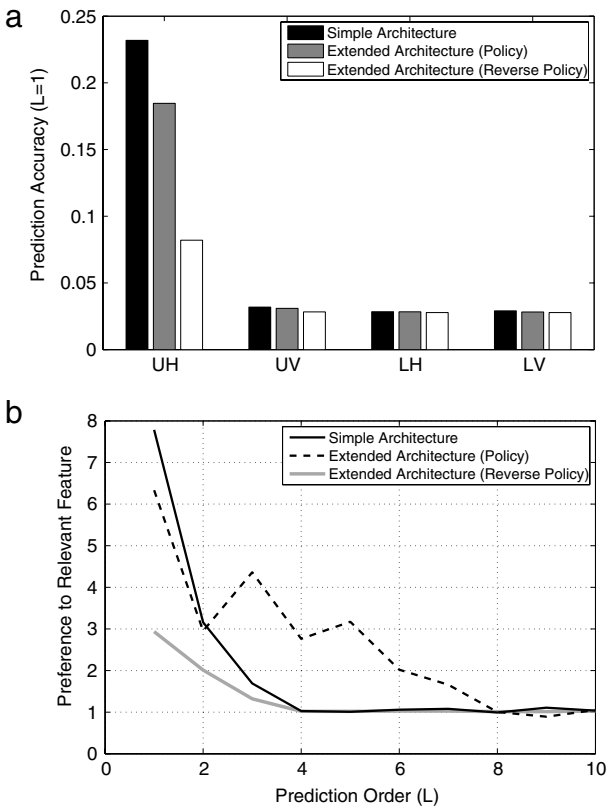

Fig. 6. (a) Prediction accuracies for upper horizontal (UH), upper vertical (UV), lower horizontal (LH) and lower vertical (LV) features, calculated for the first-order prediction task. (b) Preference to the relevant feature (UH) versus prediction order (L). For all cases the preference approaches 1 for overly long-lasting trials, meaning that for high prediction orders the networks do not show preference to predict the relevant feature.

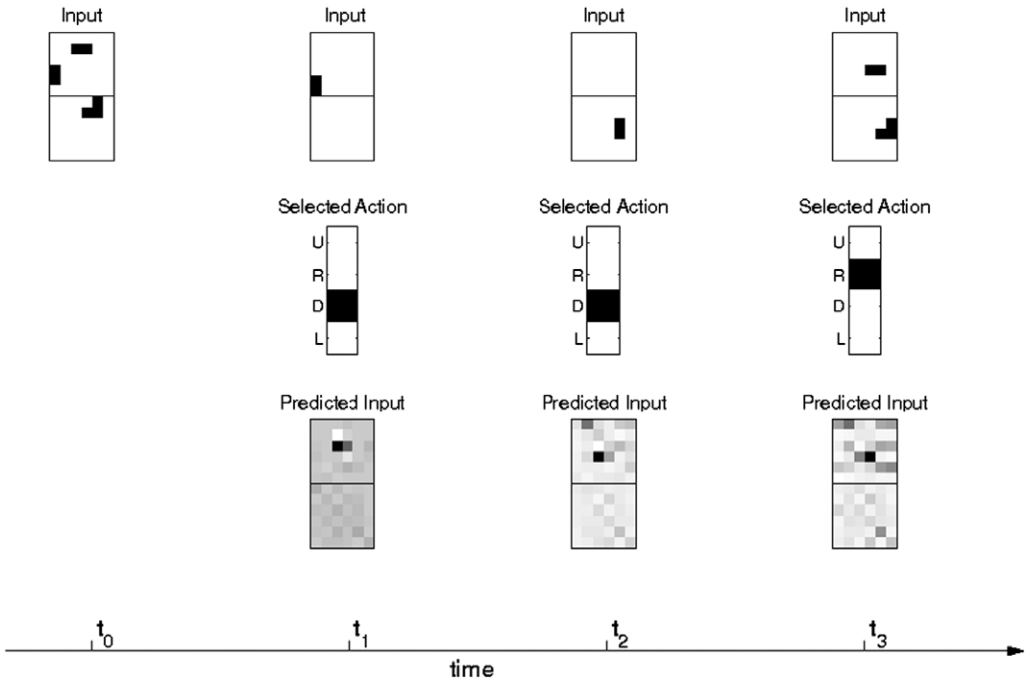

Fig. 5. Predicted network input compared to real input for three successive time steps that lead to the goal (the final position of the upper horizontal bar is the goal position). Note that features are randomly switched off, and the network tries to predict the current position of the relevant feature regardless of other features position. The action space consists of going up (U), down (D), right (R) and left (L).

that predicts the experimental results in terms of the reaction time to task-determining cues. However, it only models an instructed task-switching behavior, where the current task is assigned to the subject/model, and the only degree of freedom is how the subject/model responds to the instruction. Further investigation is required to find out how reward can modulate such task-switching mechanisms. For instance, in the context of our study, we can define multiple tasks such that in each task the reward is delivered for one of the four bars. Each task may exploit a task-specific network, one of which was presented in this study. A context-sensitive reinforcement learning approach like that of Balkenius and Winberg (2004) can be used for this purpose.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research within the "Bernstein Focus: Neurotechnology" through research grant 01GQ0840, by EU projects "PLICON" and "IM-CLeVeR", and by the Hertie Foundation. We would like to thank Arthur Franz for helpful discussions on involving the context layer. We also thank Hyundo Kim, Daniel Krieg, and Daniela Pamplona for useful comments on the document.

References

Bakker, B. (2002). Reinforcement learning with long short-term memory. *Advances in Neural Information Processing Systems*, 1475–1482.
Balkenius, C., & Winberg, S. (2004). Cognitive modeling with context sensitive reinforcement learning. In *Proceedings of AILS04*.
Downing, K. L. (2009). Predictive models in the brain. *Connection Science*, *21*(1), 39–74.
Doya, K. (1999). What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, *12*, 961–974.
Elman, J. L. (1990). Finding structure in time. *Cognition Science*, *14*, 179–211.
Graybiel, A. M. (2008). Habits, rituals, and the evaluative brain. *Annual Review of Neuroscience*, *31*, 359–387.
Haber, S., Fudge, J., & McFarland, N. (2000). Striatonigrostriatal pathways in primates form an ascending spiral from the shell to the dorsolateral striatum. *Journal of Neuroscience*, *20*(6), 2369–2382.
Hasselmo, M. E., Wyble, B. P., & Fransen, E. (2003). In M. A. Arbib (Ed.)., *The handbook of brain theory and neural networks* (2nd ed.) (pp. 761–765). Cambridge, Massachusetts: The MIT Press.
Hyman, S. E., Malenka, R. C., & Nestler, E. J. (2006). Neural mechanisms of addiction: The role of reward-related learning and memory. *Annual Review of Neuroscience*, *29*, 565–598.
Jog, M., Kubota, Y., Connolly, C., Hillegaart, V., & Graybiel, A. (1999). Building neural representations of habits. *Science*, *286*, 1745–1749.
Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (Eds.). (1995). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.
Kennerley, S. W., & Wallis, J. D. (2009). Reward-dependent modulation of working memory in lateral prefrontal cortex. *Journal of Neuroscience*, *29*(10), 3259–3270.
Krichmar, J. L. (2008). The neuromodulatory system: A framework for survival and adaptive behavior in a challenging world. *Adaptive Behavior*, *16*(6), 385–399.
Laurent, P. A. (2008). The emergence of saliency and novelty responses from reinforcement learning principles. *Neural Networks*, *21*, 1493–1499.
MacDonald, A. W., Cohen, J. D., Stenger, A., & Carter, C. S. (2000). Dissociating the role of the dorsolateral prefrontal and anterior cingulate cortex in cognitive control. *Science*, *288*(5472), 1835–1838.
Merian, N., Kessler, Y., & Adi-Japha, E. (2008). Control by action representation and input selection (caris): A theoretical framework for task switching. *Psychological Research*, *72*, 473–500.
Monsell, S. (2003). Task switching. *Trends in Cognitive Sciences*, *7*(3), 134–140.
Montague, P. R., Dayan, P., & Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *Journal of Neuroscience*, *16*, 1936–1947.
Montague, P. R., Hyman, S. E., & Cohen, J. D. (2004). Computational roles for dopamine in behavioural control. *Nature*, *431*, 760–767.
Mushiake, H., Saito, N., Sakamoto, K., Itoyama, Y., & Tanji, J. (2006). Activity in the lateral prefrontal cortex reflects multiple steps of future events in action plans. *Neuron*, *50*, 631–641.
Nagayoshi, M., Murao, H., & Tamaki, H. (2006). A state space filter for reinforcement learning in POMDPs—application to a continuous state space. In *SICE-ICASE International Joint Conference*.
Nguyen, M. (2006). Cooperative coevolutionary mixture of experts. A neuro ensemble approach for automatic decomposition of classification problems. *Doctoral dissertation*. University of Canberra, Australia (Unpublished).
Rauschecker, J. P., & Singer, W. (1979). Changes in the circuitry of the kitten visual cortex are gated by postsynaptic activity. *Nature*, *280*, 58–60.
Redgrave, P., & Gurney, K. (2006). The short-latency dopamine signal: A role in discovering novel actions? *Nature Reviews. Neuroscience*, doi:10.1038/nrn2022.
Schoups, A., Vogels, R., Qian, N., & Orban, G. (2001). Practising orientation identification improves orientation coding in V1 neurons. *Nature*, *412*, 549–553.
Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, *80*, 1–27.
Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*, 1593–1599.
Shuler, M., & Bear, M. (2006). Reward timing in the primary visual cortex. *Science*, *311*, 1606–1609.
Smallwoo, R. D., & Sondik, E. J. (1973). Optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, *21*(5), 1071–1088.
Sprague, N., & Ballard, D. (2003). Eye movements for reward maximization. In *Advances in neural information processing systems*. MIT Press.
Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
Thrun, S. B. (1992). The role of exploration in learning control. In D. A. White, & D. A. Sofge (Eds.), *Handbook for intelligent control: Neural, fuzzy and adaptive approaches*. Florence, Kentucky 41022: Van Nostrand Reinhold.
Weber, C., & Triesch, J. (2009). Goal-directed feature learning. In *International joint conference on neural networks* (pp. 3319–3326).
Whitehead, S. D., & Lin, L.-J. (1995). Reinforcement learning of non-Markov decision processes. *Artificial Intelligence*, *73*, 271–306.

Sohrab Saeb is a Ph.D. student of Computational Neuroscience at the Frankfurt Institute for Advanced Studies (FIAS), Goethe University, Frankfurt am Main, Germany. His research interests are reinforcement learning, synaptic plasticity in neuronal networks, and active vision systems.

Cornelius Weber is a Junior Fellow at the Frankfurt Institute for Advanced Studies (FIAS), Goethe University, Frankfurt am Main, Germany. His research interests are in computational neuroscience, such as vision, reinforcement learning and neuro-robotics.

Jochen Triesch is a Senior Fellow at the Frankfurt Institute for Advanced Studies (FIAS), Goethe University, Frankfurt am Main, Germany. His research interests span neural computation, human and machine vision, cognitive robotics, and models of human cognitive development.