Deep recurrent neural networks for self-organising robot control

Simón C. Smith[1], **Richard Dharmadi**[1], **Bailu Si**[2], **J. Michael Herrmann**[1]
[1]IPAB, School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, U.K.
[2]Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

Abstract

The proposed architecture applies the principle of predictive coding and deep learning in a brain-inspired approach to robotic sensorimotor control. It is composed of many layers each of which is a recurrent network. The component networks can be spontaneously active due to the homeokinetic learning rule, a principle that has been studied previously for the purpose of self-organised generation of behaviour. We present robotic simulations that illustrate the function of the network and show evidence that deeper networks enable more complex exploratory behaviour.

1 Introduction

Deep neural architectures [4, 6] have meanwhile reached a level comparable to human performance in certain pattern recognition tasks [7]. Also in robotic applications, deep networks gain more and more importance, from state abstraction to seamless end-to-end control in complex repetitive tasks [8]. Moreover, it has been speculated whether deep feedforward networks can account for some aspects of information processing in the mammalian visual system [12], which is not to say that the brain *is* nothing but a collection of deep neural networks. Quite to the contrary, brain have dynamical properties that are much richer than standard deep architectures:

- Biological neural systems consist of patches of interconnected neurons which also receive re-entrant connectivity via other patches.
- These patches are hierarchically organised to enable lateral transferability and flexible compositionality of elementary behaviours.
- Sensory inputs are not only providing information for decision about actions, but they are also analysed for traces of effects of previous actions.
- Spontaneous behaviour can occur at any level of depth and may spread in either direction.
- There is little use for supervised learning.

We are proposing here an architecture that combines the undeniable strengths of deep neural networks with an approach to meet requirements of autonomous robots. In the following, we will consider first the homeokinetically controlled sensorimotor loop [3] as the basic element of the proposed system (Sect. 2). In this way, we incorporate a source of spontaneous activity. The composition of these elements in the DIAMOnD[1] architecture (Sect. 3) will thus be able to generate activity at all levels and work in a fully self-supervised way, although it is also possible to steer the system to desired behaviour by very small guiding inputs [9]. The main layout of the architecture includes a basic layer that receives information from outside world and sends actions and is expected to represent low-level features. There is a variable number of deeper layers that interact only with the neighbouring layers and which represent more abstract features that are extracted from the data through the lower layers. The architecture learns by the homeokinetic learning rule (see below) which implies that consistency between neighbouring layers is required. We will present a few experimental results in Sect. 4, and discuss the realism and performance of the architecture as well as further work in Sect. 5.

2 Homeokinetic control

The basic element of our architecture is formed by a homeokinetic controller [3]. This unsupervised active learning control algorithm shapes the interaction between a robot and its environment by updating the parameters of the controller and of an internal model. The learning rule implies a balance between

Appearing in Proceedings of the Workshop on Robust Artificial Intelligence for Neurorobotics (RAI-NR) 2019, University of Edinburgh, Edinburgh, United Kingdom.

[1]Deep Integrated Architecture for sensoriMotor self-Organisation and Deliberation

predictability and sensitivity with respect to future inputs. The resulting behaviour is random yet temporally coherent and correlated across multiple degrees of freedom. The homeokinetic controller is a parametric function

$$\mathbf{y}_t = C(\mathbf{x}_t; \mathbf{C}) \tag{1}$$

of the vector $\mathbf{x}_t$ of current sensory states of the robot. It generates a vector of motor commands $\mathbf{y}_t$ in dependence on the current values of the parameter matrix $\mathbf{C}$. The update of the parameters is based on a comparison of actual inputs and their prediction by means of an internal model. This model

$$\hat{\mathbf{x}}_{t+1} = M(\mathbf{x}_t, \mathbf{y}_t; \mathbf{M}), \tag{2}$$

produces a prediction of future states $\hat{\mathbf{x}}_{t+1}$ based on the current input $\mathbf{x}_t$ or action $\mathbf{y}_t$ or both, and a parameter matrix $\mathbf{M}$. The difference between actual and estimated state defines the prediction error

$$\boldsymbol{\xi}_{t+1} = \mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}, \tag{3}$$

which gives rise to one of the two complementary objective functions, firstly the prediction error

$$\mathcal{E}_{t+1} = \|\mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1}\|^2, \tag{4}$$

which is used to adapt the parameters $\mathbf{M}$ of the internal model (2), and secondly the the *time loop error*

$$\mathbf{E}_t = \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2, \tag{5}$$

which is based on a post-diction $\hat{\mathbf{x}}_t$ of previous input $\mathbf{x}_t$ given the new input $\mathbf{x}_{t+1}$.

Using Eq. 1 we can always consider the model as a function defined on the state space

$$\psi(\mathbf{x}_t) = M(\mathbf{x}_t, C(\mathbf{x}_t)), \tag{6}$$

which together with Eq. 3 defines a dynamical system that represents the trajectory of the robot

$$\mathbf{x}_{t+1} = \psi(\mathbf{x}_t) + \boldsymbol{\xi}_{t+1}. \tag{7}$$

It is of importance in homeokinetic learning [3] to define an input shift $\boldsymbol{\eta}$ corresponding to the error $\boldsymbol{\xi}$. It is given as the argument $\boldsymbol{\eta} = \boldsymbol{\eta}_t$ that minimises

$$\|\mathbf{x}_{t+1} - \psi(\mathbf{x}_t + \boldsymbol{\eta})\| \tag{8}$$

or, if ψ is invertible, as $\boldsymbol{\eta}_t = \psi^{-1}(\psi(\mathbf{x}_t) + \boldsymbol{\xi}) - \mathbf{x}_t$. Using a Taylor expansion

$$\psi(\mathbf{x}_t + \boldsymbol{\eta}_t) = \psi(\mathbf{x}_t) + J(\mathbf{x}_t)\boldsymbol{\eta}_t + O(\|\boldsymbol{\eta}_t\|^2), \tag{9}$$

we express the prediction error (3) in linear order by

$$\boldsymbol{\xi}_{t+1} = J(\mathbf{x}_t)\boldsymbol{\eta}_t, \tag{10}$$

where

$$J = \left(\frac{\partial \psi_i(\mathbf{x})}{\partial x_j}\right) \tag{11}$$

is the Jacobian matrix of the system (7). If the inverse of J exists, we can use

$$\boldsymbol{\eta}_t = J_t^{-1}\boldsymbol{\xi}_{t+1} \tag{12}$$

in order to define the *time loop error*

$$\mathbf{E}_t = \|\boldsymbol{\eta}_t\|^2 = \boldsymbol{\eta}_t^\top \boldsymbol{\eta}_t = \boldsymbol{\xi}_{t+1}^\top (J_t J_t^\top)^{-1} \boldsymbol{\xi}_{t+1}. \tag{13}$$

The homeokinetic learning rule updates the parameter matrix $\mathbf{C}$ of the controller (1) by gradient descent

$$\Delta C_{ij} = -\varepsilon_{\mathbf{C}} \frac{\partial \mathbf{E}_t}{\partial C_{ij}}, \tag{14}$$

where C_{ij} is an element of $\mathbf{C}$ and $\varepsilon_{\mathbf{C}}$ is a learning rate.

If the representational power is of less importance than the flexibility [13], a quasi-linear system can be considered as sufficient. This is clearly the case in the present context where the representational power is achieved by a complex system that uses the current controller as an element. A linear controller

$$\boldsymbol{y}_t = C(\mathbf{x}_t) = g(\mathbf{C}\mathbf{x}_t + \mathbf{c}) \tag{15}$$

and a linear model

$$\hat{\mathbf{x}}_{t+1} = M(\mathbf{y}_t) = \mathbf{M}\mathbf{y}_t + \mathbf{m}, \tag{16}$$

does thus not limit the complexity of achievable control. The parameters of the controller and the model are now the matrices $\mathbf{C}$ and $\mathbf{M}$ resp., which are complemented by the matching bias vectors $\mathbf{c}$ and $\mathbf{m}$. In order to incorporate limitations of actions of the robot, the controller is quasilinear due to the elementwise sigmoidal function g. Because of the simple structure of Eq. 15, we can omit here the state dependency (2) and define the model M only in motor space. The model defines the dynamics (6)

$$\psi(\mathbf{x}) = \mathbf{M}g(\mathbf{C}\mathbf{x} + \mathbf{h}) + \mathbf{m} \tag{17}$$

and the Jacobian (11) can be obtained explicitly as $J(\mathbf{x}) = \mathbf{M}\mathbf{G}'\mathbf{C}$, where $\mathbf{G}'$ is a diagonal matrix with entries $g'(z_i)$. The parameter update (14) becomes

$$\Delta C_{ij} = \varepsilon_{\mathbf{C}}\, \boldsymbol{\eta}^\top J \frac{\partial J}{\partial C_{ij}} \boldsymbol{\eta}, \tag{18}$$

and analogously for the bias term $\mathbf{h}$. With $\boldsymbol{\mu} = \mathbf{G}'\mathbf{M}^\top \left(J^\top\right)^{-1} \boldsymbol{\eta}$ and $\boldsymbol{\zeta} = \mathbf{C}\boldsymbol{\eta}$ the learning rules for a linear controller with a linear model are

$$\Delta C_{ij} = \varepsilon_{\mathbf{C}} \mu_i \eta_j - 2\varepsilon_{\mathbf{C}} \mu_i \zeta_i y_i x_j \tag{19}$$

$$\Delta c_i = -2\varepsilon_{\mathbf{C}} \mu_i \zeta_i y_i. \tag{20}$$

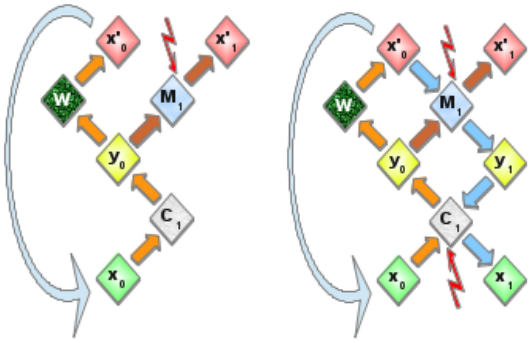

Figure 1: Schematic representation of the homeokinetic learning rule. **Left**: The prediction error in the elementary sensorimotor loop is obtained as the difference of new sensory input x'_0 and its prediction x'_1. It is used in the update of the model, see Eqs. 21, 22. **Right**: In addition, the time-loop error, i.e. the difference of previous input x_0 and re-estimated previous input x_1, is used to update the controller parameters, see Eqs. 19, 20. The downward arrow on the left merely indicates that the "new" input will serve as the input at the next time step.

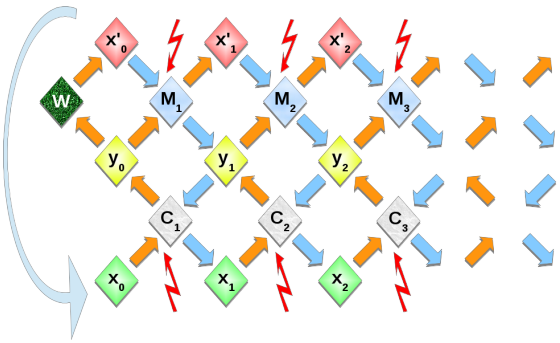

Figure 2: Homeokinetic learning in a multilayer architecture. Several instances of the homeokinetic sensorimotor loop are stacked to form a multilayer neural network. The internal model of any lower layer serves as the "world" for the next higher layer. Likewise, estimates for input obtained at by a lower layer are the inputs for the higher layers, so each layer reproduces the elementary loop shown in Fig. 1.

Simultaneously, but possibly with a different learning rate, the parameters $\mathbf{M}$ of the linear model (16) are updated via gradient descent on the standard prediction error (Eq. 4, rather than Eq. 13).

$$\Delta M_{ij} = -\varepsilon_{\mathbf{M}} \frac{\partial \mathcal{E}}{\partial M_{ij}} = \varepsilon_{\mathbf{M}} \xi_i y_j \tag{21}$$

$$\Delta b_i = -\varepsilon_{\mathbf{M}} \frac{\partial \mathcal{E}}{\partial b_j} = \varepsilon_{\mathbf{M}} \xi_i \tag{22}$$

where $\varepsilon_{\mathbf{M}}$ is the learning rate for the adaptation of the internal model. The ratio of the two learning rates $\varepsilon_{\mathbf{C}}$ and $\varepsilon_{\mathbf{M}}$ is known to be critical for the behaviour of controlled robot [13]. For the architecture presented next, an optimised ratio is to be used, see also Fig. 5.

3 The DIAMOnD model

3.1 Deep homeokinesis

The DIAMOnD model is a generalisation of the homeokinetic controller described in Sect. 2. As shown in Figs. 2 – 4, the paring of a state variable and its estimate is now extended by estimates of estimates etc. where each pair of layers corresponds to a homeokinetic controller that acts onto the lower layers as its environment and receives biases from the higher layers. As in the inner layers, the external information becomes less dominant, we can identify a formal symmetry between the actual relation between motor actions and sensory effects and the representation of this relationship in the model.

3.2 Simple variant

Layers are indicated by $\ell = 0, 1, \ldots, L$. The main structure, see Fig. 2, is described by the following equation for the controller for $\ell < L$

$$\mathbf{y}_\ell(t) = C_{\ell+1}(\mathbf{x}_\ell(t)) = g(\mathbf{C}_{\ell+1}\mathbf{x}_\ell(t) + \mathbf{c}_{\ell+1}) \tag{23}$$

with no controller for $\ell = L$. The controller update is here the same as for the one-layer model. The linear model for $\ell < L$ is given by

$$\hat{\mathbf{x}}_\ell(t+1) = M_\ell(\mathbf{y}_{\ell-1}(t), \mathbf{y}_\ell(t)) \tag{24}$$

$$= \mathbf{M}_\ell \mathbf{y}_{\ell-1}(t) + \mathbf{M}'_\ell \tilde{\mathbf{y}}_\ell(t) + \mathbf{m}_\ell. \tag{25}$$

where the $\mathbf{M}'$ matrix is updated in the same way as the $\mathbf{M}$ matrix. For $\ell = L$, this is simply

$$\hat{\mathbf{x}}_\ell(t+1) = M_\ell(\mathbf{y}_{\ell-1}(t), \mathbf{y}_\ell(t)) \tag{26}$$

$$= \mathbf{M}_\ell \mathbf{y}_{\ell-1}(t) + \mathbf{m}_\ell.$$

Note that the virtual action $\tilde{\mathbf{y}}_\ell(t)$, $\ell \geq 1$, is given by $\mathbf{x}'_{\ell+1}(t-1)$ is not the same as $\mathbf{y}_\ell(t)$ in the homeokinetic update of the controller (23), i.e. not the backpropagated value of $\mathbf{x}'_{\ell-1}(t-1)$, but instead the forward-propagated value of $\mathbf{x}'_\ell(t-1)$: First $\mathbf{x}'_\ell(t-1)$ is copied to $\mathbf{x}_\ell(t)$ for all ℓ just like in the first layer the previous sensory input becomes the new sensory input $\mathbf{x}'_0(t-1) \to \mathbf{x}_0(t)$. Then, using the controller $C'_{\ell+1}$ the virtual action $\tilde{\mathbf{y}}_\ell(t)$ is generated, which finally contributes to the prediction via Eq. 24.

3.3 Main variant

The variant with extra connections (Fig. 3) has for the controller

$$\begin{aligned} \boldsymbol{y}_\ell(t) &= C_{\ell+1}(\mathbf{x}_\ell(t)) \qquad (27)\\ &= g\big(\mathbf{C}_{\ell+1}\mathbf{x}_\ell(t)+\mathbf{C}'_{\ell+1}\mathbf{x}'_{\ell+1}(t-1)+\mathbf{c}_{\ell+1}+\mathbf{c}'_{\ell+1}\big) \end{aligned}$$

i.e. in the same way as new input $\mathbf{x}'_0(t+1)$ that is used to calculate the prediction error is also used in the next time step as input $\mathbf{x}_0(t)$, we are also for $\ell > 0$ using previous predictions as new virtual input. For the deepest layer $\ell = L$, Eq. 27 is not applied, and for the penultimate layer we have simply

$$\boldsymbol{y}_\ell(t) = C_{\ell+1}(\mathbf{x}_\ell(t)) = g(\mathbf{C}_{\ell+1}\mathbf{x}_\ell(t)+\mathbf{c}_{\ell+1}). \qquad (28)$$

For the model, Equations 24 and 26 are used as above.

While the first $\mathbf{C}$ matrix in Eq. 27 is adapted learned in the standard way (s. Eqs. 19 and 20), the matrix $\mathbf{C}'$ is updated by gradient descent with respect to the prediction error for the action

$$\mathcal{E} = (\mathbf{y}_\ell(t) - \tilde{\mathbf{y}}_\ell(t))^2,$$

where

$$\tilde{\mathbf{y}}_\ell(t) = g\left(\mathbf{C}'_{\ell+1}\mathbf{x}'_{\ell+1}(t-1)+\mathbf{c}'_{\ell+1}\right),$$

i.e. the input $\mathbf{x}'_{\ell+1}(t-1)$ from the more inner level is used to predict the motor output $\mathbf{y}_\ell(t)$. The update equations for $\mathbf{C}'$ are similar to Eqs. 21 and 22, but will contain also a derivative of g. Note that in practice $(\mathbf{M}')^{-1}$ and $\mathbf{C}'$ may be converge to a similar result, as both aim to map $\mathbf{x}'_\ell$ to $\mathbf{y}_\ell$, although one is using a linear and the other a non-linear map.

Note that the loops in Fig. 4 are not present in the network of Fig. 3, which may not be a problem as the loops have no function (yet), and may be included later. However, it is not clear what "deliberation" could mean without these loops.

We assume that the inner (deeper) layers are updated first. The deepest layer $\ell = L$ has no variables, just the controller and the model. According to Eqs. 28 and 26, no higher-level input variables are needed in order to update the variables at $\ell = L-1$.

In this way, virtual actions and virtual inputs are available to be used in Eqs. 27 and 24 to update the next layer towards the outer side, i.e. with lower ℓ. For the update of the matrices $\mathbf{M}$, $\mathbf{M}'$, $\mathbf{C}$ and $\mathbf{C}'$ the time order is not essential, if the variables are calculated as described above.

3.4 Main variant with deep associations

In a third variant, a standard deep network is connecting the x_ℓ (Fig. 4). In this case a separate matrix

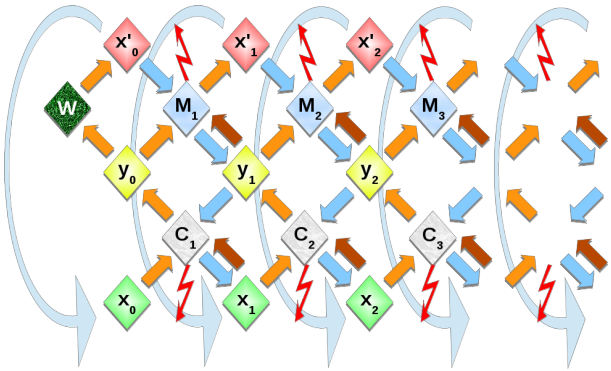

Figure 3: Same as Fig. 2, now top-down effects enabled by additional connections. This includes virtual actions analogous to the initiation of actions in the environment. The activities are propagated alternatingly through the upwards (orange and brown) arrows and through the downwards arrows (cyan), both of which correspond to a set of parallel fibres, whereas the adaptive interconnections are maintained in the controller ($\mathbf{C}$ nodes) or the model ($\mathbf{M}$ nodes).

$\mathbf{P}_\ell$ is learned for the x_ℓ which is now determined from $x_{\ell-1}$ by the connections $\mathbf{P}_{\ell+1}$ through the sigmoidal non-linearity q:

$$\begin{aligned} \mathbf{y}_\ell(t) &= C_{\ell+1}(\mathbf{x}_\ell(t)) \qquad (29)\\ &= g\big(\mathbf{C}_{\ell+1}\mathbf{x}_\ell(t)+\mathbf{C}'_{\ell+1}q(\mathbf{P}_{\ell+1}\mathbf{x}_\ell(t))+\mathbf{c}_{\ell+1}+\mathbf{c}'_{\ell+1}\big) \end{aligned}$$

$\mathbf{C}'$ is updated in the same way as $\mathbf{C}$, but also $\mathbf{P}$ and the derivative of q will occur in the learning rule. The weights $\mathbf{P}$ are learned by the activations $\mathbf{x}_\ell$ that arise due to the activations of the network. The matrices $\mathbf{R}$ in Fig. 4 play the same role as $\mathbf{P}$, but for the predicted sensor values.

It may be possible to use also the cycles in Fig. 4 more explicitely for learning, but we want to restrict ourselves to one-step learning rule, i.e. gradients are calculated only over one step of the dynamics.

4 Experimental results

4.1 Active response by the recurrent network

As a first test, we have considered the simple variant of the architecture (see Sect. 3.2 and Fig. 2) when it is driven with a sinusoidal input and the "world" reproduces simply a noisy version of the motor action as next input to the robot. Typical results are shown in Fig. 5 for a two combinations of the learning rates $\varepsilon_{\mathbf{C}}$ (19, 20) and $\varepsilon_{\mathbf{M}}$ (21, 22), which lead either to an abstracted reproduction of the input in the deeper

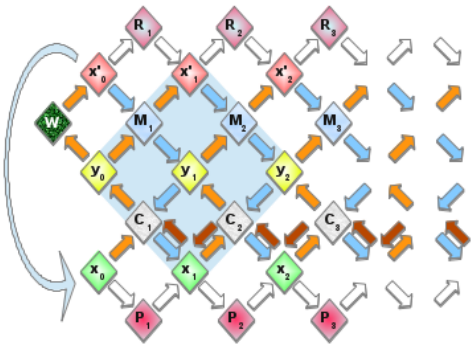

Figure 4: The network can sustain persistent activity that represents an action perception cycle. Only simples cycles are show. Activity in these subnetworks arises by self-amplification of noise or spurious activity. The full model also includes perceptual pathways consisting of bridges between input-related units. In this way the network activity becomes shaped by standard deep feed-forward networks.

layers or to a self-organisation of activity that, however remains without effect in this simple variant. At lower learning rates (left column), even deeper layers respond to the original input. In this case, the internal layers are square versions of the original input. For larger learning rates (right column), the internal layers have a different response. The fifth row shows a combination of homeokinetic adaptation (the red line between 310 and 320 sec) and noisy output while still following the input from the first layer. Deeper layers (lower rows), tend have a decay in the generation of motor action attributed to the squashing function.

4.2 A wheeled robot in the hills

The main variant (Sect. 3.3) is used in an exploration task, where a four-wheeled robot is expected to cover a large portion of an unknown territory [13]. In Fig. 6 is shown that more layers improve the exploration or rather reduce the intervals where the robots is trapped in trivial or repetitive behaviour, see Ref. [5].

4.3 A spherical robot in an arena

We also studied a simulated spherical robot which is controlled by three masses that a movable along internal axes, see Fig. 7, top. Although a more systematic study is yet to be performed, it is already obvious that adding a small number of additional layers increases the behavioural repertoire of the robot.

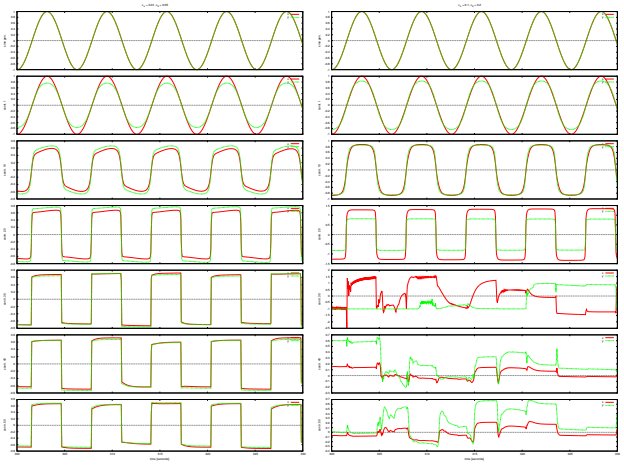

Figure 5: Activity evolution in a perceptually connected network structure, see Fig. 2. The sensory trajectory is shown by the solid line (red) and the intermediate motor action by the dashed line (green). The top row gives the input activity, the second row the activity of the first layer and the following rows show every 10th layer of the architecture to a total depth of 50. The left panel is for learning rates $\varepsilon_{\mathbf{M}} = 0.01$, $\varepsilon_{\mathbf{C}} = 0.05$, and the right one for $\varepsilon_{\mathbf{M}} = 0.1$, $\varepsilon_{\mathbf{C}} = 0.2$. While at low leaning rates, the input is similar across all layers, for larger ratios $\varepsilon_{\mathbf{M}}/\varepsilon_{\mathbf{C}}$ the model is more flexible and the deeper activity becomes largely independent on the input, which can be used (see Fig. 4).

5 Discussion

The numerical results seem to imply that a few layers are sufficient, i.e. a larger number of layers does not lead to further improvements or may require a much longer learning time than attempted here. It should, however, be considered that the tasks and environments are all very simple, such that it is not possible to generalise this observation to more complex situations. It can nevertheless be expected that the spontaneous internal activations that were observed for suitable learning rate ratios, lead to a learning time that is approximately linearly increasing with the number of layers, and not much worse. This is suggested by earlier results with homeokinetic learning rule [10].

The present model is a representation of the idea (see e.g. [1]) that it is difficult to define a clear boundary between brain and body or even between body and world. At all layers the system follow the same principles in its adaptation of the actions onto lower layers and in the learning of a model that affects higher layers. The reduction of complexity of the internal dynamics towards higher layers is counterbalanced by the autonomous activity that such that the main eigenvalue at each layer will hover near unity [11].

Terrain	Coverage rate			
$\rho = 0.25$	84	91	94	80
$\rho = 0.75$	67	71	82	74

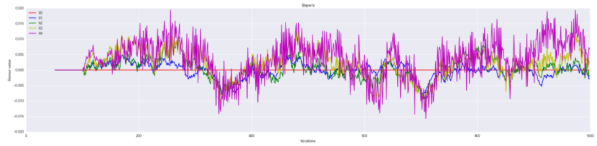

Figure 6: A four-wheeled robot exploring a hilly landscape. The table shows the percentage of area visited by the robot within 20 minutes (See Ref. [13] for details on the task). Two levels of difficulty (linear scaling ρ of the slopes) and four depths of the network (ℓ =1, 3, 5, 7) are considered, showing an increased exploration capability. The bottom graph shows the decaying prediction error over all layers (ℓ =7) which shows that the deeper layers remain unused in this task.

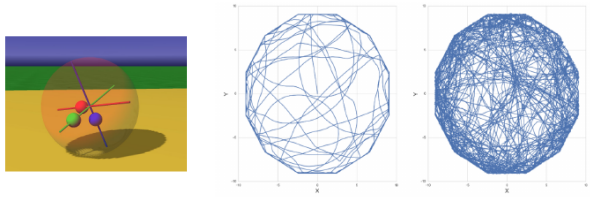

Figure 7: Spherical robot (left) in a dodecagonal arena. For a one-layer architecture, the robot mostly follows the wall (middle), while for a 3-layer network, the robot shows a highly exploratory behaviour (right).

Although the activity is updated here in parallel in all layers, the stacked structure is clearly similar to the subsumption architecture [2] as it allows for shorter or longer processing loops. It remains to be studied whether more general architectures are beneficial, especially when more complex tasks are considered.

In Figs. 2 – 4 it is understood that the dynamical variables (x, y and x') exist each in two instances, one updated by the controlling and predictive pathways, the other by the feedback within the re-estimation system. The need to disambiguate these units points to an interesting parallel to the roles of the layers of the mammalian cortex.

Finally, it should be remarked the principle of predictive coding is inherent in the architecture from the homeokinetic principle. Activity can only travel to the deeper layers if it is not already predicted by the internal model of the current layer. In some cases this can lead to a complete decay of the activity in the deeper layers (see Fig. 6), although more complex robots and more challenging environments need to be studied in order to precisely identify parallels to the predictive coding principle in natural neural systems.

Acknowledgement

We thank Georg Martius (Tübingen) and Alessandro Treves (Trieste) for stimulating discussions. J.M.H. is grateful to CAS SIA for their kind hospitality.

References

[1] M. L. Anderson, M. J. Richardson, and A. Chemero. Eroding the boundaries of cognition: Implications of embodiment. *Topics in Cognitive Science*, 4(4):717–730, 2012.

[2] R. A. Brooks. A robust layer control system for a mobile robot. *J. Robotics Automation*, RA-2:14–23, 1986.

[3] R. Der. Self-organized acquisition of situated behaviors. *Theory in Biosci.*, 120:179–187, 2001.

[4] K. Fukushima and S. Miyake. Neocognitron: Self-organizing network capable of position-invariant recognition of patterns. In *Proc. 5th Int. Conf. Patt. Recogn.*, volume 1, pages 459–461, 1980.

[5] J. M. Herrmann. Dynamical systems for predictive control of autonomous robots. *Theory in Biosci.*, 120(3-4):241–252, 2001.

[6] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *J. Machine Learning*, 17(1):1334–1373, 2016.

[9] G. Martius and J. M. Herrmann. Variants of guided self-organization for robot control. *Theory in Biosci.*, 131(3):129–137, 2011.

[10] G. Martius, J. M. Herrmann, and R. Der. Guided self-organisation for autonomous robot development. In Almeida e Costa and Francesco, editors, *Adv. in Artific. Life 9th Europ. Conf. (ECAL)*, LNCS, pages 766–775. Springer, 2007.

[11] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *preprint arXiv:1312.6120*, 2013.

[12] T. Serre, A. Oliva, and T. Poggio. A feedforward architecture accounts for rapid categorization. *PNAS*, 104(15):6424–6429, 2007.

[13] S. C. Smith and J. M. Herrmann. Evaluation of internal models in autonomous learning. *IEEE Transact. Cogn. Developm. Syst.*, 10.1109, 2018.