frontiers in Neurorobotics

BRIEF RESEARCH REPORT
published: 15 September 2020
doi: 10.3389/fnbot.2020.00062

The DIAMOND Model: Deep Recurrent Neural Networks for Self-Organizing Robot Control

Simón C. Smith[1], *Richard Dharmadi*[1], *Calum Imrie*[1], *Bailu Si*[2,3] *and J. Michael Herrmann*[1,2*]

[1] Institute of Perception, Action and Behaviour (IPAB), School of Informatics, University of Edinburgh, Edinburgh, United Kingdom, [2] State Key Laboratory of Robotics, Shenyang Institute of Automation, Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang, China, [3] School of Systems Science, Beijing Normal University, Beijing, China

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-organized generation of behavior. We present robotic simulations that illustrate the function of the network and show evidence that deeper networks enable more complex exploratory behavior.

Keywords: deep neural networks, autonomous learning, homeokinesis, self-organizing control, robot control

1. INTRODUCTION

Deep neural architectures (Fukushima and Miyake, 1980; Hinton et al., 2006) have reached a level comparable to human performance in certain pattern recognition tasks (Krizhevsky et al., 2012). Also in robotic applications, deep networks gain more and more importance, from state abstraction to seamless end-to-end control in complex repetitive tasks (Levine et al., 2016). Moreover, it has been speculated whether deep feed-forward networks can account for some aspects of information processing in the mammalian visual system (Serre et al., 2007), which is not to say that the brain *is* nothing but a collection of deep neural networks. Quite to the contrary, the brain is known to 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.
- Spontaneous behavior can occur at any level of depth and may spread in either direction.
- Sensory inputs are not only providing information for decision about actions, but are also analyzed for effects of previous actions.
- A hierarchical organization enables lateral transferability and flexible compositionality.
- There is little use for supervised learning.

Based on these considerations, we propose here an architecture that combines the undeniable strengths of deep neural networks with *homeokinesis* (Der, 2001), an approach to meet requirements of autonomous robots (see section 2). Our work connects to (Carvalho and Nolfi, 2016) where the introduction of flexibility and plasticity in a neural controller showed a good effect in a cleaning task, however, mainly based on an evolutionary approach, whereas we aim at a more principled architecture that achieves an increased flexibility by a hierarchy of identical controllers. The autonomously generate activity of higher-lever controllers provide an

OPEN ACCESS

Edited by:
Christian Tetzlaff,
University of Göttingen, Germany

Reviewed by:
Luca Patanè,
University of Catania, Italy
Yinyan Zhang,
Jinan University, China

***Correspondence:**
J. Michael Herrmann
michael.herrmann@ed.ac.uk

Received: *31 May 2020*
Accepted: *03 August 2020*
Published: *15 September 2020*

Citation:
Smith SC, Dharmadi R, Imrie C, Si B and Herrmann JM (2020) The DIAMOND Model: Deep Recurrent Neural Networks for Self-Organizing Robot Control. Front. Neurorobot. 14:62. doi: 10.3389/fnbot.2020.00062

intrinsic motivation (Oudeyer et al., 2007) for the lower ones. In this way, we are able to propose a more brain-like architecture which implicitly realizes a predictive coding principle, compare (Adams et al., 2013) for a related approach, at least in some parameter range, as discussed below. An early interesting comparison is provided by (Rusu et al., 2003) which presents a neuro-fuzzy controller for determining the behavior of a robot in a navigation task. Their architecture had a similarly layered structure, although the behaviors had to be predefined at a time when homeokinesis (Der, 2001) was just being developed. More recently, differential Hebbian learning was used to explore possible behaviors of a robot (Pinneri and Martius, 2018), presenting a more brain-like approach at the low level, whereas we aim a model that captures characteristics of the area-level organization of the brain.

In the following, we will consider first the homeokinetically controlled sensorimotor loop (Der, 2001) as the basic element of the proposed system (section 2). In this way, we incorporate a source of spontaneous activity. The composition of these elements in the DIAMOND (Deep Integrated Architecture for sensoriMotor self-Organization aNd Deliberation) architecture (section 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 behavior by very small guiding inputs (Martius and Herrmann, 2011). 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 neighboring 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 neighboring layers is required. We will present a few experimental results in section 4, and discuss the realism and performance of the architecture as well as further work in section 5.

2. HOMEOKINETIC CONTROL

The basic element of our architecture is formed by a homeokinetic controller, which we will describe here only briefly, details can be found in (Der and Martius, 2012). This unsupervised active learning control algorithm shapes the interaction between a robot and its environment by updating the parameters of a controller and of an internal model. The learning goal can be characterized as a balance of predictability and sensitivity with respect to future inputs. The resulting behavior is random yet coherent both temporally and across multiple degrees of freedom. The 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 the sensitivity of the distance between inputs and their predictions 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 that are relevant here, 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 *time loop error*

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

which is based on a post-diction $\check{\mathbf{x}}_t$ of previous input $\mathbf{x}_t$ obtained via the inverse of Equation (2) given the new input $\mathbf{x}_{t+1}$, i.e., $\mathbf{E}_t$ is calculated only at time step $t+1$, and is related to the prediction error (4) by

$$\mathbf{E}_t = \|\boldsymbol{\eta}_t\|^2 = \boldsymbol{\eta}_t^\top \boldsymbol{\eta}_t = \boldsymbol{\xi}_{t+1}^\top (J_t J_t^\top)^{-1} \boldsymbol{\xi}_{t+1}. \tag{6}$$

where J is the linearization of the maps from current input to next input dependent on the current controller. As only the projection $\boldsymbol{\eta}$ of J^{-1} on $\boldsymbol{\xi}$ is relevant, the time loop error can be efficiently estimated. The homeokinetic learning rule updates the parameter matrix $\mathbf{C}$ of the controller (1) by gradient descent

$$\Delta C_{ij} = -\varepsilon_{\mathbf{C}} \frac{\partial \mathbf{E}_t}{\partial C_{ij}}, \tag{7}$$

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 (Smith and Herrmann, 2019), then a simple quasi-linear system can be considered as sufficient. Below, when we will consider a multi-layered system, the representational power is meant to be achieved by the interaction between the layers each of which will consist of one instance of the current controller-predictor unit. A pseudo-linear controller, i.e., a quasi-linear function of the inputs with coefficients that are adaptive on the behavioral time scale,

$$\boldsymbol{y}_t = C(\mathbf{x}_t) = g(\mathbf{C}\mathbf{x}_t + \mathbf{c}) \tag{8}$$

and a linear model

$$\hat{\mathbf{x}}_{t+1} = M(\mathbf{y}_t) = \mathbf{M}\mathbf{y}_t + \mathbf{m}, \tag{9}$$

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

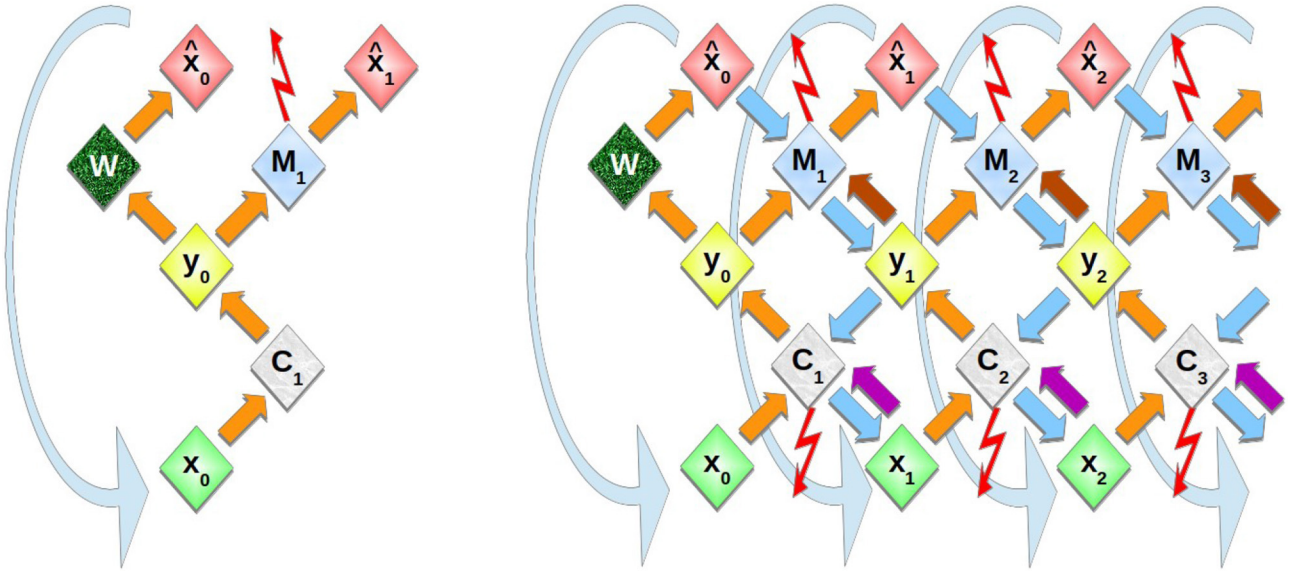


FIGURE 1 | Schematic representation of multi-layer homeokinetic learning. **Left**: In the elementary sensorimotor loop, a control action $\mathbf{y}_0$ is calculated by the controller C_1 and executed in the environment W which then produces the new input $\hat{\mathbf{x}}_0$. The prediction error is obtained as the difference of new sensory input $\hat{\mathbf{x}}_0$ and its prediction $\hat{\mathbf{x}}_1$ that was obtained from the previous input $\hat{\mathbf{x}}_0$. It is used in the update of the model M, see Equations (13), (14). **Right**: In homeokinetic learning, the time-loop error, i.e., the difference of previous input $\mathbf{x}_0$ and re-estimated previous input $\mathbf{x}_1$ (which is obtained via the downwards arrows and corresponds to $\hat{\mathbf{x}}_t$ in Equation 5), is used to update the controller parameters, see Equations (11), (12). The curved downward arrows indicate the time step: The "new" input that was previously predicted or obtained from the environment, is now used by the controller to produce the next action (rather than the re-estimated input). The inner layers follow exactly the same dynamics based on predictions from the respective outer layers rather than based on the environment. Top-down effects are included by additional connections This includes virtual actions (arrows from $\mathbf{y}_l$ to M_l) analogous to the initiation of actions in the environment, and virtual states taken into account by the controller (arrows from $\mathbf{x}_l$ to C_l). The activities are propagated alternatingly through the upwards (orange, violet, and brown) arrows and through the respective transposed matrices via downwards arrows (cyan), both of which correspond to a set of parallel fibers, whereas the adaptive interconnections are maintained in the controller (**C** nodes) or the model (**M** nodes).

of the robot, the controller is quasi-linear due to the element-wise sigmoidal function g. Because of the simple structure of Equation (8), we can omit here the state dependency (2) and define the model M only in motor space. The parameter update (7) becomes

$$\Delta C_{ij} = \varepsilon_{\mathbf{C}}\, \boldsymbol{\eta}^{\top} J \frac{\partial J}{\partial C_{ij}} \boldsymbol{\eta}, \tag{10}$$

and analogously for the bias term $\mathbf{c}$. 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

$$\Delta C_{ij} = \varepsilon_{\mathbf{C}}\mu_i\eta_j - 2\varepsilon_{\mathbf{C}}\mu_i\zeta_i y_i x_j \tag{11}$$

$$\Delta c_i = -2\varepsilon_{\mathbf{C}}\mu_i\zeta_i y_i. \tag{12}$$

Simultaneously, but possibly with a different learning rate, the parameters $\mathbf{M}$ of the linear model (9) are updated via gradient descent on the standard prediction error (Equation 4, rather than Equation 6).

$$\Delta M_{ij} = -\varepsilon_{\mathbf{M}}\frac{\partial \mathcal{E}}{\partial M_{ij}} = \varepsilon_{\mathbf{M}}\xi_i y_j \tag{13}$$

$$\Delta m_i = -\varepsilon_{\mathbf{M}}\frac{\partial \mathcal{E}}{\partial b_j} = \varepsilon_{\mathbf{M}}\xi_i \tag{14}$$

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 behavior of controlled robot (Smith and Herrmann, 2019). For the architecture presented next, an optimized ratio is to be used, see also **Figure 2**.

3. THE DIAMOND MODEL

3.1. Deep Homeokinesis

The DIAMOND model is a generalization of the homeokinetic controller described in section 2. As shown in **Figure 1**, the comparison of a state variable $\mathbf{x}(t)$ and its estimate $\hat{\mathbf{x}}(t)$ is now repeated also for estimates of estimates etc., $\mathbf{x}_0(t) = \mathbf{x}(t)$, $\mathbf{x}_1(t) = \hat{\mathbf{x}}(t)$, $\mathbf{x}_2(t), \ldots$, where each pair of neighboring layers corresponds to a homeokinetic controller that acts onto the lower layer as its environment and receives biases from the higher layer. In the inner layers (larger ℓ) the external information becomes less and less dominant.

In order to use homeokinetic learning in a multilayer architecture, several instances of the homeokinetic sensorimotor loop are stacked. 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 **Figure 1**.

3.2. Simple Variant

The architecture consists of controllers for each layer $\ell < L$ (no controller for $\ell = L$)

$$\mathbf{y}_\ell\,(t) = C_{\ell+1}\,(\mathbf{x}_\ell\,(t)) = g\,(\mathbf{C}_{\ell+1}\mathbf{x}_\ell\,(t) + \mathbf{c}_{\ell+1}) \tag{15}$$

and linear models that are given by

$$\begin{aligned}\hat{\mathbf{x}}_\ell\,(t+1) &= M_\ell(\mathbf{y}_{\ell-1}\,(t)\,,\mathbf{y}_\ell\,(t)) \\ &= \mathbf{M}_\ell\mathbf{y}_{\ell-1}\,(t) + \tilde{\mathbf{M}}_\ell\tilde{\mathbf{y}}_\ell\,(t) + \mathbf{m}_\ell.\end{aligned} \tag{16}$$

which simplifies for the top layer $\ell = L$ where $\tilde{\mathbf{y}}_L\,(t) \equiv 0$, i.e., no higher effects are present.

In Equation (16) also the effect of virtual actions $\tilde{\mathbf{y}}_\ell\,(t)$, $\ell \geq 1$ is included as follows: First, the previous prediction of a layer $\hat{\mathbf{x}}_\ell\,(t-1)$ is copied into the input unit $\mathbf{x}_\ell\,(t)$ at the beginning of the new time step, see **Figure 1**. The back-propagated input $\check{\mathbf{x}}_\ell\,(t-1)$ that was used in Equations (5) and (6) is no longer needed. From $\mathbf{x}_\ell\,(t)$ a virtual action $\mathbf{y}_\ell\,(t)$ is computed that then contributes additively to the prediction (16). The controller update is here the same as for the one-layer model, and the $\tilde{\mathbf{M}}$ matrix (not shown in the figures) is updated in the same way as the $\mathbf{M}$ matrix.

3.3. Main Variant

The variant with extra connections (**Figure 1**) has for the controller

$$\begin{aligned}\boldsymbol{y}_\ell\,(t) &= C_{\ell+1}\,(\mathbf{x}_\ell\,(t)) \\ &= g\big(\mathbf{C}_{\ell+1}\mathbf{x}_\ell\,(t) + \tilde{\mathbf{C}}_{\ell+1}\hat{\mathbf{x}}_{\ell+1}(t-1) + \mathbf{c}_{\ell+1}\big)\end{aligned} \tag{17}$$

i.e., in the same way as new input $\hat{\mathbf{x}}_0\,(t+1)$ 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 $\ell = L$, Equation (17) is not applied, and for the penultimate layer we have simply

$$\boldsymbol{y}_\ell\,(t) = C_{\ell+1}\,(\mathbf{x}_\ell\,(t)) = g\,(\mathbf{C}_{\ell+1}\mathbf{x}_\ell\,(t) + \mathbf{c}_{\ell+1})\,. \tag{18}$$

For the model, Equation (16) is used as above.

While the first $\mathbf{C}$ matrix in Equation (17) is adapted learned in the standard way (see Equations 11 and 12), the matrix $\tilde{\mathbf{C}}$ is updated by gradient descent with respect to the prediction error for the action

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

where

$$\tilde{\mathbf{y}}_\ell\,(t) = g\left(\tilde{\mathbf{C}}_{\ell+1}\hat{\mathbf{x}}_{\ell+1}\,(t-1) + \tilde{\mathbf{c}}_{\ell+1}\right),$$

i.e., the input $\hat{\mathbf{x}}_{\ell+1}\,(t-1)$ from the more inner level is used to predict the motor output $\mathbf{y}_\ell\,(t)$. The update equations for $\tilde{\mathbf{C}}$ are similar to Equations (13) and (14), but also contains a derivative of g. Note that no loops are present in the network of **Figure 1**, 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 Equation (18), 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 Equations (17) and (16) to update the next layer toward the outer side, i.e., with lower ℓ. For the update of the matrices $\mathbf{M}$, $\tilde{\mathbf{M}}$, $\mathbf{C}$ and $\tilde{\mathbf{C}}$ the time order is not essential, if the variables are calculated as described above.

3.4. Main Variant With Deep Associations

As a further variant, which is, however, not implemented in the present simulations, a standard deep neural network can be employed to connecting the inputs x_ℓ directly between neighboring levels. In this case a separate set of connections $\mathbf{P}_\ell$ would be learned for map from $x_{\ell-1}$ to x_ℓ. The weights $\mathbf{P}$ are learned by the activations $\mathbf{x}_\ell$ that arise due to the activations of the network. In addition it is possible to add a further set of connections $\mathbf{R}$ that play the same role as $\mathbf{P}$, but for the predicted sensor values.

The network can sustain persistent activity that represents an action perception cycle. Activity in the subnetworks that are completed by recurrent connections arises by self-amplification of noise or spurious activity following the homeokinetic learning of the respective controller. It may be possible to use also the cycles more explicitly for learning, but we want to restrict ourselves here to one-step learning rule, i.e., gradients are calculated only over one 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.

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 section 3.2) 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 **Figure 2** for a two combinations of the learning rates $\varepsilon_\mathbf{C}$ (11, 12) and $\varepsilon_\mathbf{M}$ (13, 14), which lead either to an abstracted reproduction of the input in the deeper layers or to a self-organization 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 s) 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.

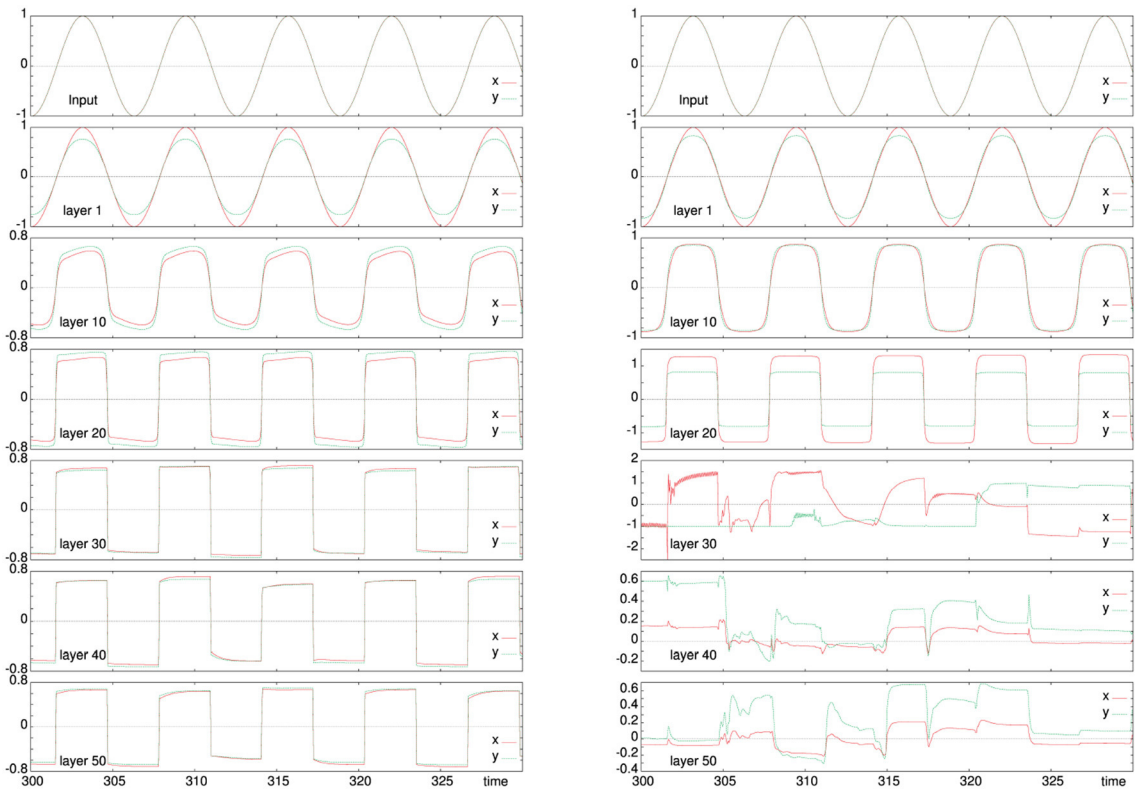


FIGURE 2 | Activity evolution in a perceptually connected network structure according to the model in section 3.2. The sensory trajectory is shown by the solid line (red) and the intermediate motor action by the dashed line (green). The top row gives the input activity, the second row the activity of 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 allows for self-organized activity in the deeper layers that is not immediately affecting the outside world.

4.2. A Wheeled Robot in the Hills

The main variant (section 3.3) is used in an exploration task, where a four-wheeled robot is expected to cover a large portion of an unknown territory (Smith and Herrmann, 2019). The hilly landscape shown left in **Figure 3** can be scaled in vertical direction such that different levels of difficulty can be achieved ranging from a flat ground (level 0) to slopes that require maximal motor power (level 1) and that can cause instabilities and thus large prediction errors (4). The activity decays in a five-layer DIAMOND model for a flat arena, as the inner layers are not needed, whereas for a hilly landscape (difficulty level > 0) the inner layers did not show much attenuation. The behavior of the robot is evaluated based on a 10×10 grid overlaid to the square-shaped arena. The number of visited grid cells is averaged over five runs for each difficulty and each controller depth and represented as a coverage rate. The total coverage was in all cases below 50% such that the increase of the coverage with time was nearly linear.

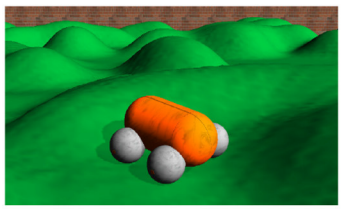
Whereas a single layer can achieve a similar performance across all terrain difficulties, for increasing difficulty of the task the higher layer are more and more engaged and take advantage of the increased errors in the terrain that provide thus a potential for a more comprehensive coverage of the arena per time unit.

4.3. A Spherical Robot in a Polygonal Arena

Finally, we studied a simulated spherical robot which is controlled by three masses that a movable along internal axes, see **Figure 4**, left. The robot is exploring freely in an polygonal environment which was chosen to discourage circular movement along the wall. The controller picks up quickly a suitable rhythm of the internal weights that is effecting in moving the robot in any direction. Collisions with wall usually stop the robot until the emergence of a different mode of the movements of the internal weights moves the robot in a different direction. Although a more systematic study is yet to be performed, it is already obvious that adding a small number of additional layers increases the behavioral repertoire of the robot and reduces the duration of any wall collisions and re-emergence of behavior in the robot. The example is also meant to demonstrate, that the applications of the learning rule and architecture are beyond exploration of a planar arena and can be used in order to generate and to organize elementary robotic behaviors.

5. DISCUSSION

The numerical results seem to imply that a few layers are sufficient, i.e., a larger number of layers does not lead



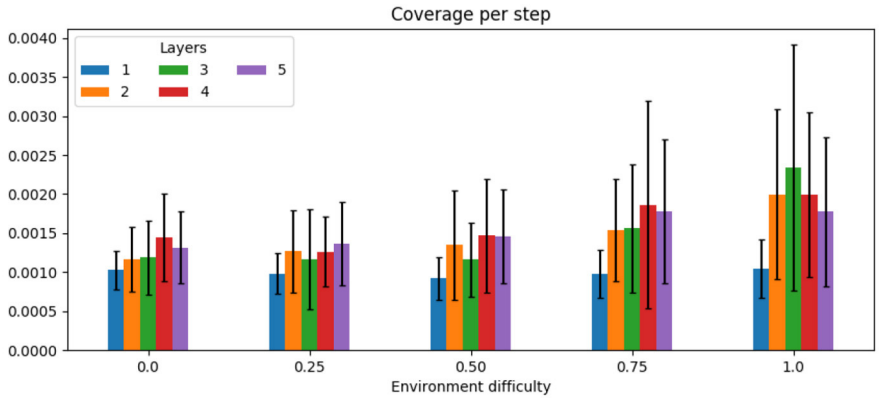


FIGURE 3 | A four-wheeled robot exploring a hilly landscape. (See text and Smith and Herrmann, 2019 for details on the task). The panel on the right shows results for five levels of difficulty (linear scaling of the slopes, with the simples level being a flat ground) and five depths of the network ($\ell = 1, 2, 3, 4, 5$) are considered, showing an increased exploration capability. The code for simulator (Der and Martius, 2012) and the DIAMOND controller architecture described here is available at https://github.com/artificialsimon/diamond.

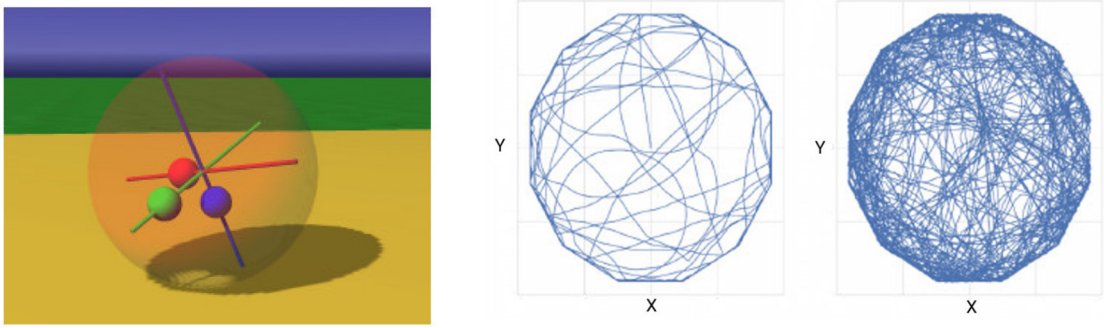


FIGURE 4 | Spherical robot **(left)** in a dodecagonal arena. For a one-layer architecture, the robot mostly follows the wall **(middle)**, while for a 3-layer network, the robot shows a highly exploratory behavior **(right)**.

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 generalize 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 (Martius et al., 2007).

The present model is a representation of the idea (see e.g., Anderson et al., 2012) 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 toward higher layers is counterbalanced by the autonomous activity such that the main eigenvalue at each layer will hover near unity (Saxe et al., 2014).

Although the activity is updated here in parallel in all layers, the stacked structure is clearly similar to the subsumption architecture (Brooks, 1986) 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 **Figure 1**, it is understood that the dynamical variables (x, y, and $\hat{x}$) 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 **Figure 3**), 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.

DATA AVAILABILITY STATEMENT

All datasets generated for this study are included in the article/supplementary material.

AUTHOR CONTRIBUTIONS

JH, BS, and SS: conception. JH and SS: model design. RD, SS, and CI: experiments. JH, BS, and SS: writing. JH: project organization.

FUNDING

SS was supported by the ORCA Hub EPSRC project (EP/R026173/1, 2017-2021). CI was supported by grant EP/L016834/1 for the University of Edinburgh, School of Informatics, Centre for Doctoral Training in Robotics and Autonomous Systems (http://www.edinburgh-robotics.org) from the UK Engineering and Physical Sciences Research Council (EPSRC). BS received funding from the National Key R&D Program of China (2019YFA0709503).

ACKNOWLEDGMENTS

We thank Georg Martius (Tübingen), Klaus Pawelzik (Bremen), Alessandro Treves (Trieste), and Hemang Kanwal (Edinburgh) for stimulating discussions. JH is grateful to CAS SIA for their kind hospitality.

REFERENCES

Adams, R. A., Shipp, S., and Friston, K. J. (2013). Predictions not commands: active inference in the motor system. *Brain Struct. Funct.* 218, 611–643. doi: 10.1007/s00429-012-0475-5

Anderson, M. L., Richardson, M. J., and Chemero, A. (2012). Eroding the boundaries of cognition: implications of embodiment. *Topics Cogn. Sci.* 4, 717–730. doi: 10.1111/j.1756-8765.2012.01211.x

Brooks, R. A. (1986). A robust layer control system for a mobile robot. *J. Robot. Automat.* RA-2, 14–23. doi: 10.1109/JRA.1986.1087032

Carvalho, J. T., and Nolfi, S. (2016). Behavioural plasticity in evolving robots. *Theory Biosci.* 135, 201–216. doi: 10.1007/s12064-016-0233-y

Der, R. (2001). Self-organized acquisition of situated behaviors. *Theory Biosci.* 120, 179–187. doi: 10.1007/s12064-001-0017-9

Der, R., and Martius, G. (2012). *The Playful Machine: Theoretical Foundation and Practical Realization of Self-Organizing Robots, Vol. 15.* Springer Science & Business Media.

Fukushima, K., and Miyake, S. (1980). "Neocognitron: self-organizing network capable of position-invariant recognition of patterns," in *Proc. 5th Int. Conf. Patt. Recogn.* (Berlin), 459–461.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.* 18, 1527–1554. doi: 10.1162/neco.2006.18.7.1527

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Lake Tahoe, NV: Curran Associates Inc.), 1097–1105.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn.* 17, 1334–1373. doi: 10.7746/jkros.2019.14.1.040

Martius, G., and Herrmann, J. M. (2011). Variants of guided self-organization for robot control. *Theory Biosci.* 131, 129–137. doi: 10.1007/s12064-011-0141-0

Martius, G., Herrmann, J. M., and Der, R. (2007). "Guided self-organisation for autonomous robot development," in *Advances in Artificial Life. ECAL 2007. Lecture Notes in Computer Science*, eds F. Almeida e Costa, L. M. Rocha, E. Costa, I. Harvey, and A. Coutinho (Berlin, Heidelberg: Springer), 766–775. doi: 10.1007/978-3-540-74913-4_77

Oudeyer, P., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* 11, 265–286. doi: 10.1109/TEVC.2006.890271

Pinneri, C., and Martius, G. (2018). "Systematic self-exploration of behaviors for robots in a dynamical systems framework," in *Artificial Life Conference Proceedings* (MIT Press), 319–326. doi: 10.1162/isal_a_00062

Rusu, P., Petriu, E. M., Whalen, T. E., Cornell, A., and Spoelder, H. J. W. (2003). Behavior-based neuro-fuzzy controller for mobile robot navigation. *IEEE Trans. Instrument. Meas.* 52, 1335–1340. doi: 10.1109/TIM.2003.816846

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *International Conference on Learning Representations*.

Serre, T., Oliva, A., and Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. *Proc. Natl. Acad. Sci. U.S.A.* 104, 6424–6429. doi: 10.1073/pnas.0700622104

Smith, S. C., and Herrmann, J. M. (2019). Evaluation of internal models in autonomous learning. *IEEE Trans. Cogn. Dev. Syst.* 11, 463–472. doi: 10.1109/TCDS.2018.2865999

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Smith, Dharmadi, Imrie, Si and Herrmann. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.