Hierarchical Exhaustive Construction of Autonomously Learning Networks

Goren Gordon [1]

Abstract. Autonomous learning is the ability to learn without external teachers. What *can* an agent learn autonomously? To answer this question we propose a hierarchical exhaustive combinatorial constructive algorithm. It generates subnetworks that attempt to learn all possible correlations between subsets of available raw data from the agent's sensors and motors. Using the concept of pruning, subnetworks that are presented with uncorrelated data sets are removed, resulting in a small pool of *viable* subnetworks. These augment the raw information dataset in higher levels, in which the exhaustive construction and pruning are repeated. The end result of the hierarchical process is a pool of viable and reliable subnetworks that represent *all* the correlations the agent can autonomously learn. One can then construct full networks by wiring learned subnetworks in order to perform specific tasks. The algorithm is implemented on a robot with a moving camera and an arm, highlighting novel concepts regarding active sensing and autonomous learning. We show that the robot's autonomously learned viable and reliable subnetworks are its sensory-motor internal models, motion detection, visual self-recognition and camera to arm coordinate transformation. The robot's only non-trivial closed-loop execution network is shown to perform reaching movements towards a moving object and is robust to noise and changes in the robot's sensors and motors due to its concurrent execution and re-learning capabilities.

1 Introduction

One of the brain's greatest virtues is its ability to learn. However, one can distinguish between two learning categories, namely, external- or teacher-mediated learning and autonomous learning, i.e. learning from internally accessible information. While naming of objects and colors is externally taught, e.g. one must be told that the word "yellow" is associated to a specific color perception, controlling your own body movements is autonomously learned [20, 5]. However, learning reaching movements are not so easy to classify [30, 2].

In this contribution we address the question: what *can* be autonomously learned, without external teachers? We wish to model our view of the brain's solution to this question. For this reason we construct a hierarchical neural network that attempts to autonomously learn *all* correlations between available data, given an agent's sensors, motors and performed actions. By *all*, we mean an exhaustive combinatorial construction of subnetworks, representing all possible subsets of available data, wherein each subnetwork *attempts* to learn a specific data subset's correlation. Many such subsets hold no correlations and are thus unlearnable; by employing the concept of pruning [26, 8], the associated subnetworks become *non-viable*, i.e. networks with no contributing neurons. The phenomenon in which an over-sized neuronal network is first initialized, followed by elimination of non-active elements is prevalent in the brain and is called exuberance [13].

Hierarchy is achieved by augmenting a higher level's available data by lower levels' viable networks. Thus, a new level exhaustively constructs subnetworks that attempt to learn correlations between outputs of lower levels' networks and raw sensory-motor data. Exuberance and pruning follows in order to distill the viable and reliable subnetworks of this level in the hierarchy. The process continues for higher hierarchical levels.

The end result of the hierarchical construction is a pool of subnetworks that represents *all* the correlations the agent can autonomously learn. This pool can then be wired in such a way so as to perform specific tasks. Since the agent cannot learn any other correlation, the combination of all possible wiring of the subnetwork pool represents the entire repertoire of tasks the agent can perform. Furthermore, since all the elements are autonomously learned, concurrent execution and learning can be performed, overcoming calibration and deterioration errors on-line.

We demonstrate the process on a real robot, with a 1 degree-of-freedom arm and a camera mounted on a single motor, representing the eye. We show that the viable subnetworks of the first level of the hierarchy represent only the internal models (IM) [14, 21, 29] of the sensory-motor coupling, among which visual motion-detection is a notable example. The second level uses the first level's subnetworks to learn more complex correlations, such as visual self-recognition [20, 5, 17]. The third and final level encompasses the entire visual field and autonomously learns visual-arm coordinate transformation [22]. The viable and reliable subnetworks are then wired to achieve the *only* functional closed-loop circuit, given the learning schedule, i.e. the only circuit that performs non-trivial action. The circuit performs a reaching task, with concurrent autonomous learning of the composing elements.

The novel features of this paper are: (i) a comprehensive brain-inspired framework of hierarchical autonomous learning of sensory-motor correlations; (ii) connection between autonomous and active sensing paradigms; (iii) a single learning algorithm that generates motion detection, self-recognition and hand-eye coordinate transformation; (iii) demonstration of a fully autonomous learning reaching robot.

The paper is organized as follows. We begin with a brief description of the model architecture and framework in Sec. 2. We then present in Sec. 3 the mathematical notations of the agent, data sets and subnetworks, followed by a description of the learning and pruning processes. Section 4 details the hierarchical exhaustive combina-

[1] Department of Neurobiology, Weizmann Institute of Science, Israel, email: goren@gorengordon.com

torial construction of all the subnetworks, followed by an analysis of the growing complexity of the network and possible execution tasks. Both sections are accompanied by a running example (Fig. 4), whose details are given in Sec. 5. Related works are decsribed in Sec.6 and the discussion in Sec. 7 concludes the paper.

2 Model Architecture and Framework

The main concept behind the proposed architecture, Fig. 1, is autonomous learning of sensory-motor correlations. The implemented algorithm is *internally supervised* learning, i.e. supervised learning where a "labeled" training set is provided by the agent itself. This is not a form of unsupervised learning [34, 25], but rather learning to predict correlations between subsets of available information. This information is the time-series of raw data from the sensors and motors of the agent, Fig. 1(a).

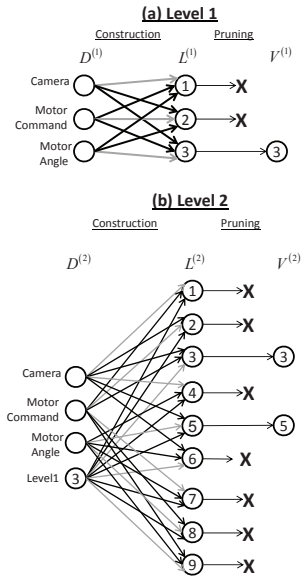

Figure 1. Model architecture, where each subnetwork (numbered circle) autonomously learns the correlation between two inputs (black arrows) and one output (gray arrows). (a) Level 1 construction of *all* subnetworks $L^{(1)}$, given camera sensor and a motor information $D^{(1)}$; followed by pruning that leaves only one subnetwork viable, $V^{(1)}$. (b) Level 2 construction of *all* subnetworks $L^{(2)}$ that include level 1 viable network; pruning leaves only subnetworks 3 and 5 viable, $V^{(2)}$.

The architecture construction is exhaustive in the sense that correlations between all subsets of available information are learned. We are interested in a brain-like architecture and thus employ a prevalent phenomenon in the brain called exuberance [13], which describes a rapid growth of connectivity between many neurons on many levels, followed by deterioration of unused or non-correlated connections. In our implementation, each subnetwork is an artificial neural network, initialized with many hidden neurons, followed by pruning [26, 8] of neurons whose decaying weights are smaller than a given threshold. If no correlations are present, the pruning process will result in a *non-viable* subnetwork, i.e. a network with no contributing neurons, thus exemplifying the network's inability to learn the subset's correlation. In some situations, usually for large-input subnetworks, pruning still results in a viable subnetwork, but it is *unreliable*, i.e. its prediction error even on the training set is high, thus exhibiting another form of inability to learn.

Following the brain's hierarchical structure, our proposed architecture is hierarchical in the sense that higher level subnetworks learn correlations between lower levels' subnetwork outputs and the raw sensory-motor information. This is reminiscent of cascade correlation networks [7, 16, 31] in which each new hidden layer neuron is connected to the input layer and lower-level hidden neurons. However, in our construction, each correlation learned is a whole (learned and viable) subnetwork that augments the input-space and allows learning of new correlations.

More specifically, in the algorithm's first level of the hierarchy, only *raw unprocessed* data from the sensors and motors are used in the aforementioned process, which ends with a small number of viable subnetworks, Fig. 1(a). In the next level of the hierarchy, the previous level's learned and viable subnetworks are combined with the sensory-motor data, Fig. 1(b). Another exhaustive combinatorial construction of new subnetworks is performed, where now each subset must include *at least* one learned subnetwork from the previous hierarchical level. Exuberance and pruning follows in order to distill the viable and reliable subnetworks. The process continues for higher hierarchical levels.

While hierarchical construction of unsupervised learning networks have been used on pure sensory data, e.g. images [12, 25], our construction focuses on internally supervised learning of correlations between sensory flow and motor actions. Hence, *active sensing* [4, 28], in which sensors are moved and controlled by the agent, is paramount to the understanding of the learned correlations. These represent what the agent can learn and predict about its own body and how it senses the environment in an active fashion. Thus, the agent can learn to predict an actuator's influence on its mounted sensor's information flow, as well as learn to determine the appropriate motor command that will generate a specific sensory input. These are the active sensing counterparts of the forward and inverse models, respectively [14, 21, 29].

3 Agent and Subnetwork Notations

This section introduces the basic elements of the proposed model, namely, the agent, its sensory-motor data and the subnetworks. The subnetworks' learning and pruning processes are then described. The section is accompanied by a running example of the implementation of the process on a real robot (Fig. 2), to better elucidate the finer details of the model.

3.1 Agent and Sensory-Motor Data

The agent is composed of N_M motors and N_S sensors. The former executes motor commands m_t^i, $i = 1, \ldots, N_M$ and the latter receives sensory input s_t^j, $j = 1, \ldots, N_S$. The dataset time-series is composed of copies of the motor commands, known as efference copies [18, 3], and sensory input, with possible delays: $D_t^{(1)} = \{m^i_{\tau^i_m}, s^j_{\tau^j_s} | \tau^i_m = t, t-1, \ldots, t-d^i_m; \tau^j_s = t, t-1, \ldots, t-d^j_s\}$, where d^i_m is the maximal delay of the ith motor command and d^j_s is the maximal delay of the jth sensory input. The (1) superscript denotes the first level of the hierarchy, which has $||D_t^{(1)}||$ data elements.

Example. The example agent is a LEGO Mindstorm robot (Fig. 2(a)) with $N_M = 2$ motors; the first controls an arm and the second the camera. It has $N_S = 3$ sensors, where the first two are

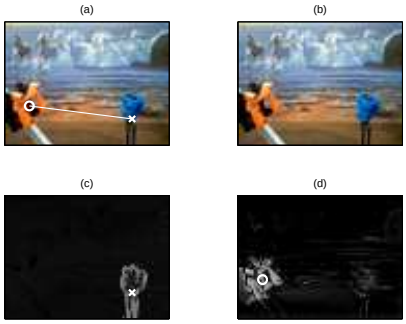

Figure 2. (a,b) Two consecutive images as captured by camera, c_t, c_{t-1}. O marks the center of the arm, X marks the center of the moving object. The line shows the distance between the two. (c) Output of motion detection subnetwork, $V_7^{(1)}$. (d) Output of visual self-recognition subnetwork, $V_1^{(2)}$ (see text for details).

proprioception sensors of the motors, $p_t^{1,2} := s_t^{1,2}$, i.e. report the motor angle, and the third is the camera, $c_t := s_t^3$. The camera is composed of 80×60 pixels, and we shall consider an increasing receptive field with increasing hierarchical levels, ranging from a single pixel in the first level, through patches of 7x7 pixels in the second level, up to the whole image in the third level of the hierarchy (see below). We consider no motor delays, $d_m^{1,2} = 0$ and a single time delay for the sensors, $d_s^{1,2,3} = 1$. Thus the data set at each time step is $D_t^{(1)} = \{m_t^1, m_t^2, p_t^1, p_{t-1}^1, p_t^2, p_{t-1}^2, c_t, c_{t-1}\}$, with $||D_t^{(1)}|| = 8$.

3.2 Subnetworks, Learning and Pruning

Subnetworks. A subnetwork is a function approximator that attempts to learn the correlation between elements of the data set. It is denoted by $L_{ijk} = p(D_i|D_j, D_k), \forall i, j, k \in D, i \neq j \neq k \neq i$, where the subscript i, j, k are indexes of the data-set elements (different from the subscript t, which indicates the current time step of the *whole* data set). We have restricted the subnetworks to a $2 \mapsto 1$ networks, i.e. only mapping of two dataset elements to another, where extensions to more elaborate mappings is straightforward. Hence, there are $||L|| = ||D|| \times (||D|| - 1) \times (||D|| - 2)/2$ possible subnetworks. The implementation of the subnetworks is via an artificial neural network (ANN), with input neurons that receive the data elements D_j, D_k, several hidden layers and output neurons that encode D_i.

Each subnetwork was taken to be multi-layered in order to allow learning of complex sensory-motor correlations. A-priori the correlation is not known, and hence initially a complex network is required for all subnetworks to allow generality. Furthermore, in this implementation there is a unique approximated output for every input element, i.e. the function approximation is deterministic, eliminating the need for probability normalization. It is also imperative, for proper comparison, that the ANN structure and parameters be the same for all subnetworks. Thus, all dataset elements were normalized to be $D \in [-1, 1]$ and saturated-linear transfer functions were used in the output neurons.

Learning. Motion of the agent's motors produces the dataset time-series, which is treated as the subnetworks' training set. Autonomous internally supervised on-line learning proceeds with the presentation of this dataset time-series to *all* the subnetworks, in parallel. We have implemented a back-propagation learning algorithm, with learning rate β and momentum α.

Pruning. Concurrent with the learning algorithm, we have implemented a pruning algorithm [26, 8] via weight decay. An additional penalty term was introduced to the ANN weights' update rule in the form of $-\gamma \text{sign}(w_{ij})$, where $0 < \gamma < 1$ is the pruning rate and w_{ij} is the respective weight. Hence, the full update rule is given by $\Delta w_{ij} = \beta \epsilon f'(x) + \alpha w_{ij} - \gamma \text{sign}(w_{ij})$, where ϵ is the (backpropagated-) error and $f(x)$ is the neuron's transfer function. For each level we have chosen γ to be proportional to the overall learning time of all networks, i.e. while learning and pruning were concurrent, as expressed in the update rule, effectively pruning manifested after learning (were possible) was stabilized.

At each time step, the sum of the absolute weights for each neuron in the hidden layers (both input and output weights) was compared to a given threshold, $h_{\text{threshold}}$; if the sum was below the threshold, that neuron was pruned. If the last neuron of a hidden layer was pruned, the subnetwork was deemed non-viable. Furthermore, for subnetworks with only two input neurons (see below), the sums of the absolute weights of the input neurons were calculated. If one sum was more than i_{mul} times greater than the other, the subnetwork was termed non-viable, since it depends only on a single input, and not both. We denote the viable subnetworks by $V \subseteq L$.

Example. Since the robot's raw dataset has $||D_t^{(1)}|| = 8$ elements, there are a total of $||L^{(1)}|| = 168$ possible subnetworks. One example of a viable subnetwork is the first motor's forward model [29] $L = p(p_t^1|p_{t-1}^1, m_t^1)$, i.e. the prediction of the next motor angle, given the previous angle and the given motor command. $L = p(p_t^1|p_{t-1}^2, c_t)$, on the other hand, is non-viable since there is no correlation between the first motor's angle, the previous other motor's angle and the current image pixels.

4 Hierarchical Construction

Using the general notations described above, we proceed to the exhaustive combinatorial construction of the subnetworks in an hierarchical fashion. We first describe the augmentation of the dataset by previous level's viable networks and then produce the current level's subnetwork pool. A complexity analysis follows, showing the double-exponential increase in network complexity, had exuberance and pruning were not implemented. The exact increase in complexity cannot be a-priori computed since it depends on the agent and its environment, but a drastic decrease in complexity results if one assumes proportional pruning. The viable and reliable subnetworks allow the wiring of specific circuits that can perform specific tasks. This is discussed at the end of the section.

4.1 Dataset Augmentation and Subnetwork Pool

The first hierarchical level dataset is composed strictly of raw sensory-motor data, $D^{(1)}$. Using these time-series as a training set, a pool of subnetworks is composed $L_{ijk}^{(1)} = p(D_i^{(1)}|D_j^{(1)}, D_k^{(1)})$, $\forall i, j, k \in D^{(1)}, i \neq j \neq k \neq i$. However, not all subnetworks are viable and following the process of concurrent learning and pruning, a subset of viable subnetworks is produced $V^{(1)} \subseteq L^{(1)}$.

The second hierarchical level now has access to the first level's viable subnetworks processed information, i.e. presented with the raw sensory-motor dataset, the viable networks produce predictions based on their learned correlations. Hence, the augmented second level dataset is $D^{(2)} = D^{(1)} \cup V^{(1)}$. One can then proceed to exhaustively construct subnetworks that will attempt to learn all possible $2 \mapsto 1$ correlations of the augmented dataset. However, only sub-

networks that include *at least* one of the previous level's viable networks will be constructed. The rest were already learned in the previous level. The second level's subnetwork pool is denoted by $L^{(2)}_{ijk} = p(D^{(2)}_i|D^{(2)}_j, D^{(2)}_k)$, $\forall i,j,k \in D^{(2)}, i \neq j \neq k \neq i$ such that $\exists D^{(2)}_{i,j,k} \in V^{(1)}$. The total number of such subnetworks is then given by $||L^{(2)}|| = ||D^{(2)}|| \times (||D^{(2)}|| - 1) \times (||D^{(2)}|| - 2)/2 - ||L^{(1)}||$. Concurrent learning and pruning then follows to produce $V^{(2)}$ viable networks.

This can be easily generalized to higher levels, as follows:

$$D^{(n)} = D^{(n-1)} \cup V^{(n-1)} \tag{1}$$

$$L^{(n)}_{ijk} = p(D^{(n)}_i|D^{(n)}_j, D^{(n)}_k), \quad \forall i,j,k \in D^{(n)}, i \neq j \neq k \neq i \quad \text{s.t.} \exists D^{(n)}_{i,j,k} \in V^{(n-1)} \tag{2}$$

$$||L^{(n)}|| = ||D^{(n)}|| \times (||D^{(n)}|| - 1) \times (||D^{(n)}|| - 2)/2 - \sum_{m=1}^{n-1} ||L^{(m)}|| \tag{3}$$

Example. The robot's first level of the hierarchy produces $||L^{(1)}|| = 168$ subnetworks. However, only internal models (IM) of the sensory-motor dataset have correlations (see below, Fig. 3). These are presented to three subnetworks that relate p^1_t, p^1_{t-1}, m^1_t (IM of the arm); three subnetworks that relate p^2_t, p^2_{t-1}, m^2_t (IM of the eye-motor) and; three subnetworks that relate c_t, c_{t-1}, m^2_t (IM of the eye-motor and camera). Hence, $||V^{(1)}|| = 9$ and $||D^{(2)}|| = 17$, resulting in $||L^{(2)}|| = 1872$.

4.2 Complexity Analysis

One can consider the increase in the number of subnetworks as the hierarchical levels grow. Assume that pruning is not implemented; this results in changing Eq. (1) to $D^{(n)} = D^{(n-1)} \cup L^{(n-1)}$. Thus, denoting $x_n := ||D^{(n)}||$ and $y_n := ||L^{(n)}||$, we get the following recursion relations:

$$x_n = x_{n-1} + y_{n-1} \tag{4}$$

$$y_n = x_n(x_n - 1)(x_n - 2)/2 - y_{n-1} \quad y_0 = 0 \tag{5}$$

This results in double exponential dependency on the hierarchy level: $y_n \sim O(x_1^{3^n})$.

However, one can counter this increase by the use of exuberance and pruning. Consider that only a very small subset of the subnetwork pool remains viable, such that $||V^{(n)}|| = \kappa||D^{(n)}||$, $\kappa > 1$. This results in a drastic decrease in subnetwork pool complexity, to a single exponential dependence: $y_n \sim O(\kappa^{3(n-1)})$.

Furthermore, as seen in the example below, there is a possibility that several of the viable networks are *equivalent*, i.e. they convey the same information and are thus redundant. This may result in a linear increase in the number of informative viable networks as hierarchical level increases. This is indeed the case in the robot example analyzed below.

4.3 Execution of a Task

Once the hierarchical levels' viable subnetworks were learned, one can construct a full network to perform specific tasks. This can be done by connecting one subnetwork's output to another's input. Notice that during execution the inputs to the subnetworks may differ from those during the learning phase. However, they must accommodate the inputs' dimensionality and content.

In order for a task to be operational, the last subnetwork must have a motor output. This drastically restricts the number of subnetworks that may reside in the end of the task network. However, the number of possible wirings cannot be a-priori computed, since the number and characteristics of the *viable* networks are not known.

Example. One of the first level's viable subnetwork is the arm's inverse model, $V^{(1)} = p(m^1_t|p^1_t, p^1_{t-1})$. During autonomous learning, it was presented with the known time-series, where p^1_{t-1} was the delayed proprioception input. However, during execution it can serve as a mechanism to achieve a specific angle position, p^1_{goal}: $V^{(1)} = p(m^1_{t+1}|p^1_{\text{goal}}, p^1_t)$, i.e. it determines the next motor command m^1_{t+1} via the goal position and the current position. Hence, it may reside at the end of a functional network.

5 Robot Implementation

We have implemented the proposed model on a LEGO Mindstorm robot (Fig. 1(a), Supp. Movie), with a 1 degree-of-freedom (DOF) arm, m^1_t, and a single motor m^2_t that controls the pan of a USB camera, c_t. As described above, the raw data set at each time step is $D^{(1)}_t = \{m^1_t, m^2_t, p^1_t, p^1_{t-1}, p^2_t, p^2_{t-1}, c_t, c_{t-1}\}$, where $p^{1,2}_t$ is the proprioception information relating the motor angle. First, the hierarchical construction of the viable subnetwork pool is presented. It is followed by a presentation of a possible wiring of the learned subnetworks to accomplish a reaching task.

5.1 Construction of Viable Subnetwork Pools

5.1.1 *First level*

The first level data set, $D^{(1)}$, was used to construct an exhaustive set of all $||L^{(1)}|| = 168$ possible subnetworks. All the subnetworks had two hidden layers with four neurons each, and $\beta = \alpha = 0.1$, $\gamma = 0.0001$. The robot moved its arm motor randomly, and its camera motor in a succession of random motion and then no motion. This was repeated for 100,000 time steps, in which concurrent learning and pruning was performed on all subnetworks in parallel, Fig. 3.

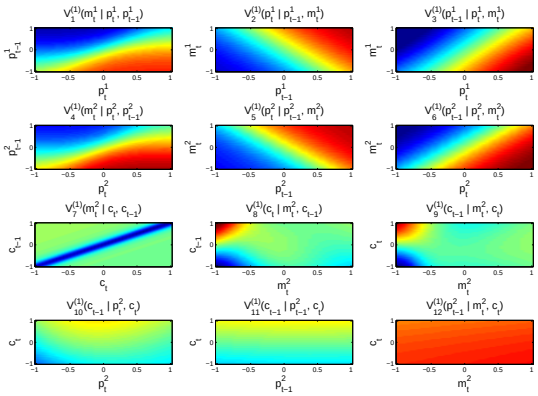

Figure 3. Mapping of viable subnetworks of the first level. In order to visualize the image-space, here c_t is taken to be the normazlized gray-scale value of the RGB pixel in $V^{(1)}_{7-12}$.

The first level of the hierarchy represents the lowest level in the visual pathway, e.g. retina. Hence, c_t in this level was set to be a single RGB pixel. During the learning process, a subnetwork that had either

c_t or c_{t-1} was presented with all the image pixels in a randomized permutation order, thus having a training set 4800-times larger than other networks. Furthermore, it means that the input/output layers of such networks had more neurons: networks that had one or two camera inputs had four or six input neurons, respectively; networks that had a camera output had three output neurons (RGB). We thus normalized the learning and pruning rate, β, γ, by scaling them by $1/4800$. We further factorized pruning by scaling the pruning rate, γ by one over the number of input neurons.

Moreover we introduced a minor modification to expedite learning: Motor commands were continuous in their regime, $m_t^{1,2} \in [-1, 1]$. However, learning visuo-motor correlation is more related to the existence of motion, rather than its direction. Learning to distinguish between $|m_t^{1,2}|$ using $m_t^{1,2}$ is very difficult and time-consuming for a small ANN, since it is non-monotonic. Hence, for all subnetworks in the first level that contained at least one visual input and at least one motor command, the latter was taken to be its absolute value. For example, $L^{(1)} = p(m_t^2|c_t, c_{t-1}) \Rightarrow p(|m_t^2||c_t, c_{t-1})$.

Viable subnetworks. The viable networks in the end of this process were (Fig. 3): $V_1^{(1)} = p(m_t^1|p_t^1, p_{t-1}^1)$, $V_2^{(1)} = p(p_t^1|m_t^1, p_{t-1}^1)$ and $V_3^{(1)} = p(p_{t-1}^1|p_t^1, m_t^1)$ representing the inverse, forward and postdiction internal models of the arm motor, respectively [29]; $V_4^{(1)} = p(m_t^2|p_t^2, p_{t-1}^2)$, $V_5^{(1)} = p(p_t^2|m_t^2, p_{t-1}^2)$ and $V_6^{(1)} = p(p_{t-1}^2|p_t^2, m_t^2)$ representing the inverse, forward and postdiction IM of the camera motor, respectively, and; $V_7^{(1)} = p(m_t^2|c_t, c_{t-1})$, $V_8^{(1)} = p(c_t|m_t^2, c_{t-1})$ and $V_9^{(1)} = p(c_{t-1}|c_t, m_t^2)$ representing the inverse, forward and postdiction IM of the camera motor and camera image, respectively. Three more networks were viable, but their corresponding prediction maps hold no information: $V_{10}^{(1)} = p(c_{t-1}|p_t^2, c_t)$,$V_{11}^{(1)} = p(c_t|p_{t-1}^2, c_{t-1})$ and $V_{12}^{(1)} = p(p_{t-1}^2|m_t^2, c_t)$. We believe that further learning would have resulted in their pruning.

The first six viable networks have been described intensively in the literature [14, 21, 29]. For a 1 DOF constellation they are very simple, whereas for more DOF there are known problems, mainly in the inverse models [14, 21]. However, this is not the main topic of the paper and thus we do not elaborate on it.

Of special interest is $V_7^{(1)} = p(m_t^2|c_t, c_{t-1})$ (Fig. 2(c)); it receives the current image, the previous image and learns whether the camera motor has moved. In a non-homogeneous visual environment, when the camera moves, most of the pixels' colors change; when the camera does not move, most of the pixels' colors do not change. Hence, this subnetwork represents an *autonomously learned* visual change detector.

Figure 3 shows the learned mapping of the viable subnetworks of the first level. As can be seen, the motor's internal models are (almost) linear mapping. $V_7^{(1)}$ is the visual change detection: as described above, the output was rescaled to be 1 for motion and -1 for no motion. This shows that if two pixels are identical, the output is -1, whereas if they are different, the output is 1, constituting a true change detector. Notice that $V_{8,9}^{(1)}$ are similar and represent the visual prediction: if there is no motion (left side), the output pixel is identical to the input pixel; if there is motion (right side), nothing can be said of the output pixel. Finally, the maps of $V_{10-12}^{(1)}$ are almost flat, representing no information or correlation.

Novel features. The nine viable networks of the first level hint towards a more general concept of autonomous learning: active sensing circuits [4, 28], whereby motor commands influence sensory information, enables autonomous learning of internal models of the sensory-motor coupling. Thus, while one interpretation of $V_7^{(1)}$ is visual change detection, it is learned similarly to $V_{1,4}^{(1)}$ which are "proper" inverse models [14, 21]. Hence, it can be used as a visual inverse model, or active vision [1, 23, 19]: given the current image and a goal image, what is the proper motor command? Conversely, $V_{1,4}^{(1)}$ can be used as joint-angle change detectors: given the current and previous angles, was the joint moved? The exhaustive construction of all subnetworks brought these novel concepts to light: (i) There is one-to-one mapping between inverse models and change detectors and; (ii) only active sensing coupling enables autonomous learning of internal models and change detectors.

5.1.2 Second level

Eqs. (1-3) result in $||D^{(2)}|| = 17$ and $||L^{(2)}|| = 1872$. This was computationally too expensive for the setup we have considered, so made the following restrictions: (i) only the arm motor was considered, m_t^1, p_t^1; (ii) no sensory or motor delays were considered, $d_m^{1,2} = d_s^{1,2,3} = 0$; (iii) only the arm motor internal models and the visual change detection were taken, $V_{1-3,7}^{(1)}$ and; (iv) only subnetworks that had at least one visual component were considered. This resulted in a total of $||L^{(2)}|| = 72$ subnetworks. While this is considerably less than the exhaustive pool, it is still large enough to demonstrate all the proposed concepts. All the subnetworks had two hidden layers with ten neurons each, and $\beta = 5, \alpha = 0.1, \gamma = 0.001$. The robot moved its arm motor randomly for 10,000 time steps, in which concurrent learning and pruning was performed on all subnetworks in parallel.

The second level of the hierarchy represents a higher level in the visual pathway, one in which features are detected [15]. Hence, c_t in this level was set to be a 7x7 RGB pixel array. As with the previous level, each time step all 7x7 arrays composing the image were presented to the subnetworks in a randomized permutation order. Similarly, the number of the input/output layers' neurons were enlarged and the learning and pruning rates were rescaled. Since all the subnetworks had at least one image component, the rescaled learning rate β was always much smaller than 1.

Viable subnetworks. Only five subnetworks were viable at the end of the process (Fig. 4), namely $V_{1-5}^{(2)} = p(V_7^{(1)}|c_t, K)$, where $K = \{m_t^1, p_t^1, V_{1,2,3}^{(1)}\}$. Three more subnetworks "survived" the learning and pruning process, but held no information: $V_6^{(2)} = p(p_t^1|m_t^1, V_7^{(1)})$, $V_7^{(2)} = p(p_t^1|c_t, V_1^{(1)})$ and $V_7^{(2)} = p(p_t^1|c_t, V_3^{(1)})$. Examining $V_{1-5}^{(2)}$ reveals that in the implemented learning schedule, i.e. only a moving arm, they are all similar; they learn the transformation from image patches, c_t, to moving objects in the visual field $V_7^{(1)}$. Since the only moving object in the training set was the arm itself, they all represent *autonomous learning* of visual self-recognition, Fig. 2(d).

Figure 4 shows the execution output of the 8 viable subnetworks of level two. As can be seen, $V_{1-5}^{(2)}$ are practically identical. While $V_6^{(2)}$ seems to hold some information, the distinction between the arm and background is extremely small.

Novel features. Other works have shown autonomous learning of visual self-recognition [20, 17, 5], yet none have done so in a hierarchical manner starting from raw sensory data. Rather, all have used intensive pre-programmed image processing prior to learning. Furthermore, while we have not used the fully exhaustive second-level subnetwork pool, due to computational hardware limitations, this mapping *emerged* as the only viable and functional mapping out of 72 others.

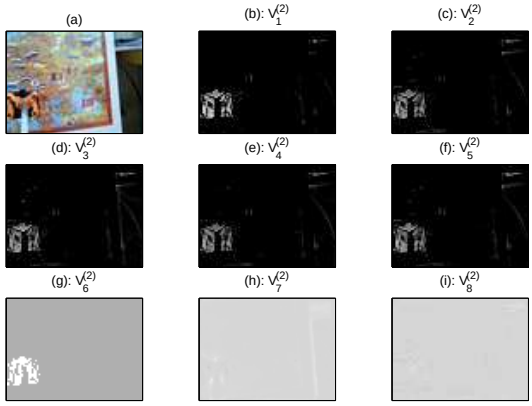

Figure 4. Execution map of the viable level 2 subnetworks.

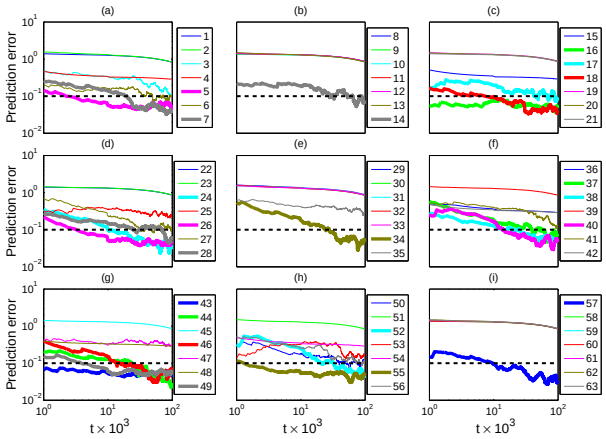

Figure 5. Prediction error of all 63 subnetworks of level 3 (see Appendix). Reliable networks are emphasized. Dashed black line delineate reliability threshold.

5.1.3 Third level

Since we have shown the equivalence of the five viable second-level subnetworks, we can augment the data set with only one, e.g. $V_1^{(2)}$. This still results in a large exhaustive pool of subnetworks, so we made the same restrictions as in the second level. Furthermore, we have introduced a modification to the execution of second level viable subnetworks whose output was a single-channel image. This is usually due to *detection* of visual objects and hence its magnitude is not important, only its sign. We therefore changed: $\text{output} \in [-1, 1] \Rightarrow \text{output} \in \{-1, 1\}$, by thresholding at 0. This only expedited the learning of the higher levels, and does not change the results of the paper. For example, the execution output $V_5^{(2)} = p(V_7^{(1)}|c_t, p_t^1)$ was set to be the sign of the output neuron.

These alterations result in $||L^{(3)}|| = 63$ subnetworks, Fig. 5 (see Appendix). Again, this is a rather small pool, but can still demonstrate the basic principles presented here. All the subnetworks had two hidden layers with ten neurons each, with parameters $\beta = 0.01, \alpha = 0.1$. The robot moved its arm motor randomly for 100,000 time steps, in which only learning was performed on all subnetworks in parallel. Since this is the last level of the hierarchy and the subnetworks had numerous input neurons (see below), pruning proved to be an inefficient mechanism for distilling viable networks. Hence we employ a prediction error threshold, $e_{\text{threshold}} = 0.1$; subnetworks whose average prediction error did not decline below it at the end of the learning stage were deemed *unreliable*, Fig. 5.

The third and last level of the hierarchy represents the highest level in the visual pathway, one that considers the entire field of view [6]. Hence, c_t in this level was set to be the entire image and in contrary to the previous two levels, a single presentation was done in each time step. Furthermore, rescaling of β was not required.

Viable subnetworks. Twenty subnetworks ended up reliable, $V_{1-20}^{(3)}$ (see Appendix). However, they have several things in common: $V_1^{(2)}$ is always an input (and not an output), and the output consists of arm parameters (not image, or change detection). From this and further analysis, one can interpret the learned correlation: it is the transformation between the visual location of the hand, via $V_1^{(2)}$, and the current arm position. Hence, this constitutes *autonomous learning* of visual-to-arm coordinate transformation.

Novel features. Previous works have described algorithms for coordinate transformations [22]. However, they employed intense image processing prior to learning. Furthermore, this is the first time, to the best of our knowledge, that the same algorithm produces motion detection, visual self-recognition and visual-to-arm coordinate transformation.

5.2 Reaching Network

We next wish to construct a *closed loop* wiring between learned subnetworks that begins with sensory information and ends up in a motor command. By *closed loop* we mean that there is no external goal or task, but that the motion of the motors are determined by the wiring itself. Hierarchical construction followed by pruning resulted in the following viable and reliable subnetworks: nine networks from the first level, five equivalent subnetworks from the second level and twenty equivalent subnetworks from the third and last level.

We focus on the motion of the arm only, hence the subnetwork at the end of the closed loop must have an output of an arm motor-related parameter, Fig. 6. The arm's inverse model, $V_1^{(1)}$ is the natural subnetwork that obeys this requirement. However, some of the third level subnetworks, $V_{13-18}^{(3)}$ (see Appendix), have the inverse model as their output and thus also obey that requirement.

The image-arm coordinate transformation subnetworks, $V_{1-12}^{(3)}$ (see Appendix) are the only non-trivial subnetworks that can serve as an input to the arm's inverse model. This is so because these subnetworks have either p_t^1 or $V_2^{(1)}$, which is the forward model of the arm, as their output. This equivalent class transforms visual self-recognition, $V_1^{(2)}$ to the arm's angle, regardless of the other input, e.g. m_t^1, c_t. Hence, these are actually a $1 \mapsto 1$ subnetworks. However, since they are third-level subnetworks, they have an input of *the whole image*, resulting in 4800 input neurons.

We focus on the arm inverse model. During execution, the inverse model receives the current position of the arm and should receive a goal position. However, since we are interested in a closed loop, the goal should come from another subnetwork. The viable candidate subnetworks are $V_{2,3}^{(1)}$ and $V_{1-12}^{(3)}$, where the former group closes a trivial loop of forward/inverse models of the same motor. The latter group are equivalent and transform a visual image to an arm coordinate, so for simplicity we choose $V_1^{(3)} = p(p_t^1|V_1^{(2)}, m_t^1)$. The wiring of $V_1^{(3)} \rightarrow V_1^{(1)}$ results in motion towards a visual object. Finally, $V_1^{(3)}$ receives a *single channel* full image as its input. Six viable

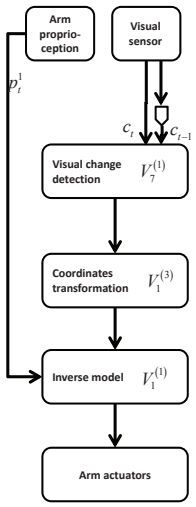

Figure 6. Reaching close-loop network, connecting viable subnetworks $V_7^{(1)} \rightarrow V_1^{(3)} \rightarrow V_1^{(1)}$.

subnetworks produce this output, namely, $V_{1-5}^{(2)}$ and $V_7^{(1)}$. The former five are equivalent, representing the arm's visual self-recognition (Fig. 2(d)) and result in trivial motion of the arm towards itself.

The latter's motion detection capabilities (Fig. 2(c)) close the loop: $(c_t, c_{t-1}) \rightarrow V_7^{(1)} \rightarrow V_1^{(3)} \rightarrow V_1^{(1)}$. In words, two consecutive images produce a change map; coordinate transformation transfers the location of detected changes into the arm's position; the arm's inverse model produces the correct arm motor command to reach towards the moving object, Fig. 6.

As mentioned above, $V_{13-18}^{(3)}$ can also be the final subnetworks of the closed loop (see Appendix). Analyzing their input/output relations show that they are equivalent and can serve as a combination of visual-to-arm coordinate transformation *and* inverse model. This means that one can replace the wiring $V_7^{(1)} \rightarrow V_{1-12}^{(3)} \rightarrow V_1^{(1)}$ described above with $V_7^{(1)} \rightarrow V_{13-18}^{(3)}$. We have not done so in the robot implementation in order to show that several subnetworks can be executed and concurrently re-learned.

We wish to emphasize that the presented execution circuit is the *only* functional non-trivial closed loop composed of viable and reliable subnetworks that operate the arm. The other possible wirings that result in motion of the arm perform trivial motions, such as moving towards the current position of the arm, i.e. not moving. While we have introduced learning schedule restrictions in the second and third levels, the process has still demonstrated extreme convergence: from a total of $168+72+63 = 303$ initial subnetworks, exuberance, pruning and reliability reduced the pool to a mere $9+5+20 = 34$ viable and reliable subnetworks, which can produce a single *emergent* functional closed loop, namely, reaching a moving target.

5.3 Concurrent Execution and Learning

Since all the components of the reaching network can be autonomously learned, the network possesses a unique characteristic: it can continuously and autonomously re-learn all its elements. We have shown that the inputs to the viable subnetworks are different in the learning and execution phases. Hence, during execution of the reaching motion, re-learning requires another "pass" through the learning network. Thus, for example, during concurrent learning, the coordinate transformation subnetwork $V_1^{(3)}$ should receive its input from the self-recognition subnetwork, $V_1^{(2)}$, and not the motion detection one $V_7^{(1)}$, as in the execution phase. This requires the maintenance and re-learning of subnetworks that do not actually contribute to the reaching task, e.g. $V_1^{(2)}$.

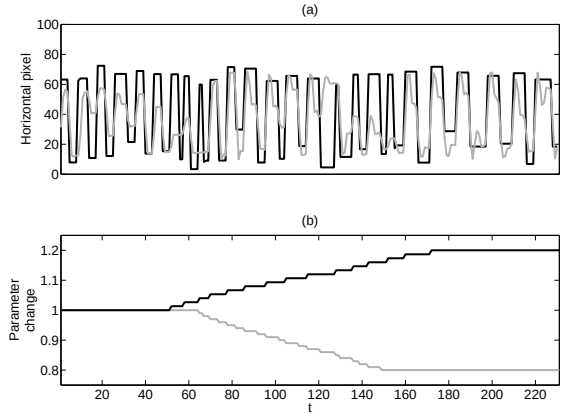

Figure 7. Reaching a moving object with concurrent learning (see Supp. Movie). (a) Horizontal position of moving object (black) and arm (gray) as a function of time steps. (b) Percent change of motor (black) and camera (gray) parameters.

Furthermore, we have implemented a specific arm and camera motion schedule to improve learning during the learning phase: motion detection was learned quickly because the camera consecutively moved and then rested; arm visual self-recognition and coordinate transformation were learned when only the arm moved. This means that during the reaching execution stage, some biases could be introduced. For example, if motion detection were learned only when the *arm* moved, it would re-learn that only arm features contribute to motion detection and would not average out all possible pixel combinations to produce a true motion detector. Hence, in the sequence demonstrated in Fig. 7(a), motion detection was not re-learned, due to the camera's immobility. However, the rest of the components, namely, $V_1^{(1)}, V_1^{(2)}$ and $V_1^{(3)}$ were learned concurrently with the reaching execution.

In the testing of the entire reaching network, we have added another LEGO Mindstorm motor that controlled a moving object, Fig. 2(a,blue object;c). It was moved to a random position to the left and then right of the image where it then made a sudden move, Fig. 7(a, black). The reaching network was then executed and monitored via the camera, Fig. 2(a,red hand;d). We have also introduced gradual changes to the robot's sensors and motors to ascertain its concurrent execution and re-learning capabilities.

Figure 7(a) shows the horizontal location of the two detected entities, namely, arm and object and nicely demonstrates how the arm follows the movement of the object. The figure first shows initial calibration errors, corrected after 70 times steps (see Supp. Movie). It is followed by an introduction of a slow change in the motor plant of the arm, whose maximal power increased by 20% and a degradation of the red channel of the camera to 80% of its value, Fig. 7(b). As can be seen in Fig. 7(a), the reaching motion is again accurate after the change due to the network's concurrent re-learning capabilities.

6 Related Works

In [5] an information theoretic approach was taken in order to autonomously learn what a robot can control. By using mutual information and moving its own hand, the robot could discern its hands and then its fingers. However, the model included image processing in order to extract the relevant information measures.

Ref. [30] explores several models of learning how to reach, whose main features are the learning of internal models, namely, forward and inverse kinematics and dynamics of the arm. While the suggested model autonomously learns these internal models by moving the arm, it does not address the issue of arm self-recognition, but rather uses intense image processing to acquire the external coordinates of the arm and the objects it interacts with.

Another humanoid robot was used to autonomously learn the coordinates transformation between the head and the arm [22]. This was done by using a fixed gaze by which the head was turned to keep the hand in the center of the image. Then the head and arm proprioception angle information was used to autonomously learn the coordinates transformation.

Work of the same group [20] has also implemented learning of the self via periodic motions of the hand and finding the corresponding image features. Furthermore, building saliency maps of the visual image and interacting with the environment, enabled building object models in the vicinity of the robot.

Building hierarchical learning networks that extract higher order correlations in the data was also performed by several groups [12, 25]. However, they have focused on image processing, while the model presented here focuses more on the interaction between the motorized action and sensors, both visual and proprioception. Ref. [33] learns generalized value functions which indeed relate states and actions, but does not perform an exhaustive search over these possibilities. Another related model is that of Hierarchical Temporal Memory [9], which learns temporal sensory information in different hierarchies, where the time-scales change with hierarchy level. The model presented here focuses more on the sensory-*motor* correlations with the emphasis of creating a functional executable circuit from the learned and viable subnetworks. This novel focus highlights the relevance of active sensing [4, 28] to the autonomous learning paradigm.

In the current implementation, we have used random motion in order to learn the sensory-motor correlations. However, many works have implemented active learning concepts to expedite such learning (see [10] and references therein). More specifically, the concept of intrinsic reward that originates from learning these transformation is a promising avenue of research [24, 32, 11] and will be explored in future work.

7 Discussion

A brain-inspired novel algorithm implementing the prevalent concept of exuberance [13] was introduced: it starts by constructing an exhaustive pool of subnetworks and then during learning prunes away those that are presented with uncorrelated data sets. Repeating the procedure in a hierarchical manner results in a pool of *viable* and *reliable* subnetworks that represent *all* the correlations the agent can autonomously learn. These serve as an alphabet to construct executable circuits that perform specific tasks, with concurrent autonomous learning of all composing elements.

The complexity analysis presented above can hint to the underlying cause of exuberance in the brain [13]. Initially, the organism does not know which sensory information correlates with which motor command. Hence, starting with an exhaustive connectivity and then pruning away the non-functional elements in a hierarchical manner results in reduced hierarchical complexity.

We have focused on $2 \mapsto 1$ correlations and not all-to-all correlations, as in self-organizing maps [27], in order to show the emergence of specific functional networks, e.g. motion detection and self-recognition. Had we constructed the first level network to have all the inputs and one output, e.g. the camera motor command $p(m_t^2|D^{(1)})$, it would not have learned motion detection $p(m_t^2|c_t, c_{t-1})$, since the camera motor inverse model $p(m_t^2|p_t^2, p_{t-1}^2)$ would have better predicted the motor command. One may thus conclude that dividing the input information to subsets is beneficial for constructing a truly exhaustive map of learnable transformations.

Furthermore, the architecture presented here used a large initial network and then utilized pruning. One may have opted for starting with a small network and increasing it via, e.g. cascade correlation networks [7, 16, 31]. However, since there is evidence for death of inactive neurons, but less of increase in the number of neurons in the brain, we believe that pruning connections and non-active neurons is more suitable to our brain-inspired framework than adding neurons to a network. A thorough comparison of the performance of the two options is beyond the scope of this paper.

The proposed model and its implementation give rise to several interesting aspects of autonomous learning in general. From a neurobiological perspective, the specific suggested architecture for learning how to reach has a novel prediction that suggests connectivities that are mandatory in order to achieve learning, e.g. an efference copy of the eye muscles must reach the first motion detection station, which is as low as the retina. However, it seems unlikely that basic change detection is a learned quality of the nervous system, since many (if not all) low-level neurons have that characteristic. How can this be resolved? First, the question we ask is fundamental: what can be learned, expanding beyond what indeed is learned in biological systems. Second, evolution may also play a significant role, i.e. it is possible that lower species have a learned motion detection architecture, but higher ones developed a genetically-encoded mechanism to achieve the same goal. This has the added advantage of requiring less time to manifest, meaning an organism with a "hard-wired" motion detector will detect motion earlier in development than one that has to learn it. Furthermore, it enables the learning neuronal system to focus on more complicated sensory-motor correlations, i.e. higher loops. Third, the biological substrates that living organisms are made of have some inherent qualities, and change detection, or other simple transformations, may be one of them, thus eliminating the need to learn it during development.

Another important issue is what determines the overall execution connectivity of the viable subnetworks. A possible biological reasoning is that initially all the subnetworks are connected to each other and only those that serve some purpose and achieve a specific (rewarding) goal survive in development. Hence, one can picture a fully connected network in which all available information serve as both input and output to many correlation-learning neuronal networks. These networks are then only way-stations to other neuronal networks that serve as the second tier in the hierarchy and so on. Selecting which network will be activated and which will control the muscles at any given moment probably involves a rewarding mechanism which is beyond the scope of the present work.

The exhaustive constructive algorithm has also surfaced a novel concept relating active sensing [4, 28] and autonomous learning: in the first level of the hierarchy, *only* active sensing sensory-motor cou-

plings produce viable subnetworks and those represent internal models. Specifically, the inverse model subnetworks can be used in two complementary ways: (i) determining the proper motor command to achieve a specific sensory goal and; (ii) detecting changes in the sensory information.

Finally, the proposed implementation of the exhaustive architecture is just the first step towards a more complex network with many more capabilities. One can think of new subnetworks in all levels of the hierarchy: a network that learns to map a visual receptive field to the specific directional motion of the eye can learn orientation lines and edge detection; using two cameras, one can autonomously learn a stereoscopic coordinates transformation; head, torso and leg movements can also be autonomously learned and add more flavor to a much richer network.

To conclude, we have developed a methodology of an exhaustive hierarchical construction of autonomously learning agents. We have shown that by implementing exuberance and pruning, only viable and reliable networks that actually encode correlations in the sensory-motor information survive. Those augment higher levels' available data to introduce more complex subnetworks. We have implemented it on a robot and showed that three levels of the hierarchy can produce a completely autonomously learned reaching arm, that is robust to noise due to its concurrent execution and re-learning capabilities. We envision that this model can be implemented on virtually any robotic agent and can augment existing pre-programed algorithmic controls.

Appendix: Level 3 Subnetworks

$$
\begin{array}{ll}
L_1^{(3)} = p(V_1^{(2)}|p_t^1, m_t^1) & L_2^{(3)} = p(V_1^{(2)}|p_t^1, c_t) \\
L_3^{(3)} = p(m_t^1|p_t^1, V_1^{(2)}) & L_4^{(3)} = p(c_t|p_t^1, V_1^{(2)}) \\
L_5^{(3)} = p(V_1^{(1)}|p_t^1, V_1^{(2)}) & L_6^{(3)} = p(V_3^{(1)}|p_t^1, V_1^{(2)}) \\
L_7^{(3)} = p(V_2^{(1)}|p_t^1, V_1^{(2)}) & L_8^{(3)} = p(V_7^{(1)}|p_t^1, V_1^{(2)}) \\
L_9^{(3)} = p(V_1^{(2)}|p_t^1, V_1^{(1)}) & L_{10}^{(3)} = p(V_1^{(2)}|p_t^1, V_3^{(1)}) \\
L_{11}^{(3)} = p(V_1^{(2)}|p_t^1, V_2^{(1)}) & L_{12}^{(3)} = p(V_1^{(2)}|p_t^1, V_7^{(1)}) \\
L_{13}^{(3)} = p(V_1^{(2)}|m_t^1, c_t) & L_{14}^{(3)} = p(p_t^1|m_t^1, V_1^{(2)}) \\
L_{15}^{(3)} = p(c_t|m_t^1, V_1^{(2)}) & L_{16}^{(3)} = p(V_1^{(1)}|m_t^1, V_1^{(2)}) \\
L_{17}^{(3)} = p(V_3^{(1)}|m_t^1, V_1^{(2)}) & L_{18}^{(3)} = p(V_2^{(1)}|m_t^1, V_1^{(2)}) \\
L_{19}^{(3)} = p(V_7^{(1)}|m_t^1, V_1^{(2)}) & L_{20}^{(3)} = p(V_1^{(2)}|m_t^1, V_1^{(1)}) \\
L_{21}^{(3)} = p(V_1^{(2)}|m_t^1, V_3^{(1)}) & L_{22}^{(3)} = p(V_1^{(2)}|m_t^1, V_2^{(1)}) \\
L_{23}^{(3)} = p(V_1^{(2)}|m_t^1, V_7^{(1)}) & L_{24}^{(3)} = p(p_t^1|c_t, V_1^{(2)}) \\
L_{25}^{(3)} = p(m_t^1|c_t, V_1^{(2)}) & L_{26}^{(3)} = p(V_1^{(1)}|c_t, V_1^{(2)}) \\
L_{27}^{(3)} = p(V_3^{(1)}|c_t, V_1^{(2)}) & L_{28}^{(3)} = p(V_2^{(1)}|c_t, V_1^{(2)}) \\
L_{29}^{(3)} = p(V_7^{(1)}|c_t, V_1^{(2)}) & L_{30}^{(3)} = p(V_1^{(2)}|c_t, V_1^{(1)}) \\
L_{31}^{(3)} = p(V_1^{(2)}|c_t, V_3^{(1)}) & L_{32}^{(3)} = p(V_1^{(2)}|c_t, V_2^{(1)}) \\
L_{33}^{(3)} = p(V_1^{(2)}|c_t, V_7^{(1)}) & L_{34}^{(3)} = p(p_t^1|V_1^{(2)}, V_1^{(1)}) \\
L_{35}^{(3)} = p(m_t^1|V_1^{(2)}, V_1^{(1)}) & L_{36}^{(3)} = p(c_t|V_1^{(2)}, V_1^{(1)}) \\
L_{37}^{(3)} = p(V_3^{(1)}|V_1^{(2)}, V_1^{(1)}) & L_{38}^{(3)} = p(V_2^{(1)}|V_1^{(2)}, V_1^{(1)}) \\
L_{39}^{(3)} = p(V_7^{(1)}|V_1^{(2)}, V_1^{(1)}) & L_{40}^{(3)} = p(p_t^1|V_1^{(2)}, V_3^{(1)}) \\
L_{41}^{(3)} = p(m_t^1|V_1^{(2)}, V_3^{(1)}) & L_{42}^{(3)} = p(c_t|V_1^{(2)}, V_3^{(1)}) \\
L_{43}^{(3)} = p(V_1^{(1)}|V_1^{(2)}, V_3^{(1)}) & L_{44}^{(3)} = p(V_2^{(1)}|V_1^{(2)}, V_3^{(1)}) \\
L_{45}^{(3)} = p(V_7^{(1)}|V_1^{(2)}, V_3^{(1)}) & L_{46}^{(3)} = p(p_t^1|V_1^{(2)}, V_2^{(1)}) \\
L_{47}^{(3)} = p(m_t^1|V_1^{(2)}, V_2^{(1)}) & L_{48}^{(3)} = p(c_t|V_1^{(2)}, V_2^{(1)}) \\
L_{49}^{(3)} = p(V_1^{(1)}|V_1^{(2)}, V_2^{(1)}) & L_{50}^{(3)} = p(V_3^{(1)}|V_1^{(2)}, V_2^{(1)}) \\
L_{51}^{(3)} = p(V_7^{(1)}|V_1^{(2)}, V_2^{(1)}) & L_{52}^{(3)} = p(p_t^1|V_1^{(2)}, V_7^{(1)}) \\
L_{53}^{(3)} = p(m_t^1|V_1^{(2)}, V_7^{(1)}) & L_{54}^{(3)} = p(c_t|V_1^{(2)}, V_7^{(1)}) \\
L_{55}^{(3)} = p(V_1^{(1)}|V_1^{(2)}, V_7^{(1)}) & L_{56}^{(3)} = p(V_3^{(1)}|V_1^{(2)}, V_7^{(1)}) \\
L_{57}^{(3)} = p(V_2^{(1)}|V_1^{(2)}, V_7^{(1)}) & L_{58}^{(3)} = p(V_1^{(2)}|V_1^{(1)}, V_3^{(1)}) \\
L_{59}^{(3)} = p(V_1^{(2)}|V_1^{(1)}, V_2^{(1)}) & L_{60}^{(3)} = p(V_1^{(2)}|V_1^{(1)}, V_7^{(1)}) \\
L_{61}^{(3)} = p(V_1^{(2)}|V_3^{(1)}, V_2^{(1)}) & L_{62}^{(3)} = p(V_1^{(2)}|V_3^{(1)}, V_7^{(1)}) \\
L_{63}^{(3)} = p(V_1^{(2)}|V_2^{(1)}, V_7^{(1)}) & .
\end{array}
\tag{6}
$$

The twenty viable subnetworks can be grouped into three equivalent classes:

1. Subnetworks with arm angle-related output: $V_{1-12}^{(3)} = L_{14,24,34,40,46,52,7,18,28,38,44,57}^{(3)}$. Notice that $V_2^{(1)}$ is the forward model of the arm motor and during training is completely equivalent to p_t^1.
2. Subnetworks with arm motor-related output: $V_{13-18}^{(3)} = L_{5,16,26,43,49,55}^{(3)}$. Notice that $V_1^{(1)}$ which appears as an output in these subnetworks is the inverse model of the arm. During training it is completely equivalent to m_t^1.
3. Subnetworks with the arm's previous angle output: $V_{19-20}^{(3)} = L_{17,37}^{(3)}$. Notice that $V_3^{(1)}$ which appears the output in these subnetworks is the postdiction model of the arm and during training is completely equivalent to p_{t-1}^1.

REFERENCES

[1] D. H. Ballard, 'Animate vision', *Artif. Intell.*, **48**(1), 57–86, (1991).
[2] N. E. Berthier and R. Keen, 'Development of reaching in infancy', *Exp Brain Res*, **169**(4), 507–18, (2006).
[3] T. B. Crapse and M. A. Sommer, 'Corollary discharge across the animal kingdom', *Nat Rev Neurosci*, **9**(8), 587–600, (2008).

[4] K. E. Cullen, 'Sensory signals during active versus passive movement', *Curr Opin Neurobiol*, **14**(6), 698–706, (2004).

[5] A. Edsinger and C. C. Kemp, 'What can i control? a framework for robot self-discovery', in *The sixth international conference on epigenetic robotics (EpiRob)*.

[6] R. Epstein and N. Kanwisher, 'A cortical representation of the local visual environment', *Nature*, **392**(6676), 598–601, (1998).

[7] S.E. Fahlman and C. Lebiere. The cascade-correlation learning architecture, 1990.

[8] N. Fnaiech, F. Fnaiech, and B. Jervis, *Feedforward Neural Networks Pruning Algorithms*, 1–16, Electrical Engineering Handbook, CRC Press, 2011.

[9] D. George and J. Hawkins, 'Towards a mathematical theory of cortical micro-circuits', *PLoS Comput Biol*, **5**(10), e1000532, (2009).

[10] G. Gordon, D. M. Kaplan, B. Lankow, D. Y. Little, J. Sherwin, B. A. Suter, and L. Thaler, 'Toward an integrated approach to perception and action: conference report and future directions', *Front Syst Neurosci*, **5**, 20, (2011).

[11] G. Gordon and E. Ahissar, 'Reinforcement active learning hierarchical loops', in *International Joint Conference on Neural Networks (IJCNN)*.

[12] G. E. Hinton and R. R. Salakhutdinov, 'Reducing the dimensionality of data with neural networks', *Science*, **313**(5786), 504–7, (2006).

[13] G. M. Innocenti and D. J. Price, 'Exuberance in the development of cortical networks', *Nat Rev Neurosci*, **6**(12), 955–965, (2005).

[14] M. I. Jordan, 'Forward models: Supervised learning with a distal teacher', *Cognitive Science*, **16**, 307–354, (1992).

[15] N. Kanwisher, J. McDermott, and M. M. Chun, 'The fusiform face area: a module in human extrastriate cortex specialized for face perception', *J Neurosci*, **17**(11), 4302–11, (1997).

[16] M. Kawato, Y. Maeda, Y. Uno, and R. Suzuki, 'Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion', *Biol Cybern*, **62**(4), 275–88, (1990).

[17] C. C. Kemp and A. Edsinger, 'What can i control?: The development of visual categories for a robot's body and the world that it in uences.', in *5th IEEE International Conference on Development and Learning (ICDL5): Special Session on Perceptual Systems and their Development*.

[18] H. Lalazar and E. Vaadia, 'Neural basis of sensorimotor learning: modifying internal models', *Curr Opin Neurobiol*, (2008).

[19] E. P. Merriam and C. L. Colby, 'Active vision in parietal and extrastriate cortex', *The Neuroscientist*, **11**(5), 484–493, (2005).

[20] L. Natale, F. Orbona, G. Metta, and G. Sandini, 'Exploring the world through grasping: a developmental approach', in *IEEE International Symposium on Computational Intelligence in Robots and Automation*.

[21] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schlkopf, 'Learning inverse dynamics: A comparison', in *European Symposium on Artificial Neural Networks (ESANN 2008)*, pp. 13–18.

[22] F. Nori, L. Natale, G. Sandini, and G. Metta, 'Autonomous learning of 3d reaching in a humanoid robot', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[23] J. Kevin O'Regan, No, euml, and Alva , 'A sensorimotor account of vision and visual consciousness', *Behavioral and Brain Sciences*, **24**(05), 939–973, (2001).

[24] P. Y. Oudeyer, F. Kaplan, and V. V. Hafner, 'Intrinsic motivation systems for autonomous mental development', *Evolutionary Computation, IEEE Transactions on*, **11**(2), 265–286, (2007).

[25] M. Ranzato, Y. lan Boureau, and Y. Lecun, 'Sparse feature learning for deep belief networks', in *Advances in Neural Information Processing Systems*.

[26] R. Reed, 'Pruning algorithms-a survey', *Neural Networks, IEEE Transactions on*, **4**(5), 740–747, (1993).

[27] H. Ritter, 'Self-organizing maps for robot control', in *Proceedings of the 7th International Conference on Artificial Neural Networks* (1997).

[28] C. E. Schroeder, D. A. Wilson, T. Radman, H. Scharfman, and P. Lakatos, 'Dynamics of active sensing and perceptual selection', *Curr Opin Neurobiol*, **20**(2), 172–6, (2010).

[29] R. Shadmehr and J. W. Krakauer, 'A computational neuroanatomy for motor control', *Exp Brain Res*, **185**(3), 359–81, (2008).

[30] R. Shadmehr, 'Generalization as a behavioral window to the neural mechanisms of learning internal models', *Human Movement Science*, **23**, 543–568, (2004).

[31] S. K. Sharma and P. Chandra, 'Constructive neural networks: A review', *International Journal of Engineering Science and Technology*, **2**(12), 7847–7855, (2010).

[32] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, 'Intrinsically motivated reinforcement learning: An evolutionary perspective', *Autonomous Mental Development, IEEE Transactions on*, **2**(2), 70–82, (2010).

[33] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup, 'Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction', in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2* (2011).

[34] E. Todorov and Z. Ghahramani, 'Unsupervised learning of sensory-motor primitives', in *25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. **2**, 1750-1753, (2003).