2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Predictive Hebbian Association of Time-Delayed Inputs with Actions in a Developmental Robot Platform

Martin F. Stoelen, Davide Marocco, Angelo Cangelosi, Fabio Bonsignorio and Carlos Balaguer

***Abstract*— The work described here explores a neural network architecture that can be embedded directly in the real-time sensorimotor coordination loop of a developmental robot platform. We take inspiration from the way children are able to learn while interacting with a teacher, in particular the use of prediction of the teacher actions to improve own learning. The architecture is based on two neural networks that operate online, and in parallel, one for learning and one for prediction. A Hebbian learning rule is used to associate the high-dimensional afferent sensor input at different time-delays with the current efferent motor commands corresponding to the teacher demonstration. The predictions of future motor commands are used to limit the growth of the neural network weights, and to enable the robot to smoothly continue movements the teacher has begun. Results on a simulated iCub robot learning object interaction tasks are presented, including an analysis of the sensitivity to changes in the task setup. We also outline the first implementation on the real iCub platform.**

I. Introduction

CHILDREN learn to manipulate objects through continuous interaction with the environment and with human teachers, effectively addressing the "degrees of freedom problem" [1] in spite of initially having very limited motor skills. Much of the learning occurs during interaction, which means it can immediately be applied, and that the teacher can adapt to the progress of the child. A robot operating outside laboratory conditions similarly has to overcome the "curse of dimensionality" when learning novel manipulation tasks. That is, such tasks typically require the association of motor commands with the high-dimensional sensor input coming from the visual, auditory, tactile, and proprioceptive modalities. It may be beneficial to take inspiration from the way children develop to overcome these challenges. We believe this requires, at least, the following robot capabilities:

1) Automatically extract task-relevant information from the robot's full high-dimensional sensory input.
2) Predict future teacher actions from the observation of the dynamics of the relevant sensory input.
3) Transition smoothly from teacher-guided action to own action based on such predictions.
4) Integrate information from high-level labels with the low-level prediction, when available.
5) Perform both learning and prediction online while the human teacher is providing demonstrations.

Working towards such capabilities, we previously explored a Neural Network (NN) for associating online high-dimensional robot actions and sensor input [2]. The present paper extends this work, with two NNs operating in parallel, one for associative learning and one for motor prediction. The prediction is used for: a) limiting the growth of the synapse weights, and b) smoothly transferring control from the human teacher to the robot.

II. Related Work

One approach for representing learned robot skills is the Dynamic Movement Primitives (DMP) paradigm [3]. Here each primitive can be seen as an attractor landscape to a predefined goal, and the approach thus provides a stable trajectory from any point in state-space. However, collisions and uncertain sensor data must be handled explicitly. See for example [4], where additional sensor data was used to adapt online a previously learned DMP. Locally Weighted Learning (LWL) is another related paradigm [5]. Online multi-map regression approaches have been applied to learning a Finite State Machine (FSM) of subtasks and their policies [6]. Common to these approaches is the need to explicitly model each movement or primitive. Approaches where sets of movements can be modelled and coordinated implicitly exists, typically based on Recurrent Neural Networks (RNN). For example work on a hierarchical Reservoir Computing (RC) network for imitation learning [7], or RNNs that use parametric biases for representing behaviour patterns [8]. This also includes Multiple Time-scales Recurrent Neural Networks (MTRNN) [9].

There is also work on recognising the intent of a human user, for example online intent prediction on assistive wheelchairs [10]. Other work has focused on online recognition of manipulator actions, through a modified formulation of DMPs [11]. However, each action was encoded using an explicit model and the learning was performed offline. Online learning is likely beneficial for autonomous robots in general, and particularly during interaction with a human teacher. Incremental imitation learning approaches do exist, for example using a Gaussian Mixture Regression (GMR)

Martin F. Stoelen and Carlos Balaguer are members of the RoboticsLab research group within the Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, Spain (email: {mstoelen, balaguer}@ing.uc3m.es).

Fabio Bonsignorio is a member of the RoboticsLab as Santander Chair of Excellence in Robotics at Universidad Carlos III de Madrid and CEO/founder of Heron Robots of Genova, Italy (email: fbonsign@ing.uc3m.es or fabio.bonsignorio@heronrobots.com).

Davide Marocco and Angelo Cangelosi are with the Centre for Robotics and Neural Systems, University of Plymouth, Plymouth, UK (email: {davide.marocco, acangelosi}@plymouth.ac.uk).

This work was supported by the European Commission FP7 Project ITALK (ICT-214668) within the Cognitive Systems and Robotics unit (FP7 ICT Challenge 2) and the RoboCity2030-II-CM project (S2009/DPI-1559), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

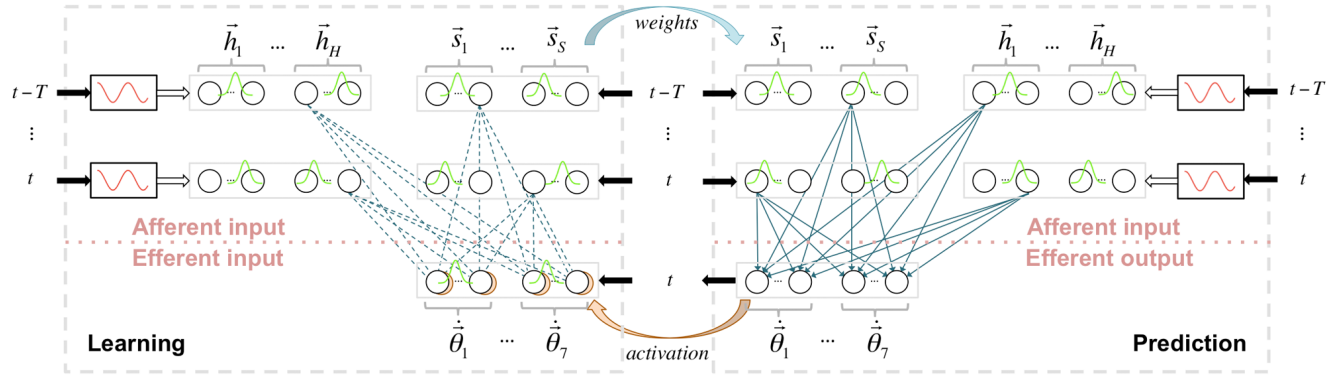


Fig. 1. The two neural networks. For clarity of presentation only the synapses for one neuron in each past layer are shown. Dashed synapses indicate learning (without propagation), solid arrow synapses indicate propagation of activity. Grey solid boxes represent layers, while the grey dashed boxes indicate the border of each neural network. Thick solid arrows represent input from the outside. Example Gaussian activation curves shown in green.

to represent each task [12]. Here the learning is performed after each demonstration, and on sensor data that has first been through a dimensionality reduction. Learning in parallel with a demonstration has been suggested as a potential application for Liquid State Machines (LSM) with online linear regression [13]. There is also related work on life-long learning in autonomous robots, for example learning about a large set of policies in parallel [14].

III. Neural Networks and Embedding in Robot

A. Introduction

The two NNs and their interrelationships can be seen in Fig. 1. Each NN has multiple layers receiving the same afferent input, and each layer has a specific time-delay. The layers therefore represent the afferent input at different moments, from the present time t to T time steps into the past (denoted $t-T$). The afferent input includes sensor data from S sensors. Each sensor is represented by a set of neurons, here written as $\vec{s}$. The jth neuron for the afferent sensor input is denoted as a_j. High-level hypotheses about the task can also be provided, with the set of neurons representing one hypothesis written as $\vec{h}$. See Section III-F for more details. The jth neuron for the afferent hypothesis input is denoted as a_j^{hyp}. Each neuron in each set is assigned a specific sensor/hypothesis value, and the sets of neurons are therefore discretizations of the sensor/hypothesis inputs. Both NNs have one layer with neurons representing the efferent commands. For the work described here the arm joint velocities were assumed as commands, with the set of neurons representing each joint velocity being denoted as $\dot{\vec{\theta}}$. The ith neuron for the efferent commands is denoted as e_i.

B. Operating Modes

The NNs have two operating modes, learning and actuation. During learning, the teacher moves the robot through the task, while the learning NN learns the associations between the time-delayed afferent input and the efferent input representing the motor commands. The prediction NN simultaneously generates motor predictions. The activations generated by the prediction NN is used in the learning, and the prediction NN receives updates on the NN weights periodically. During actuation, only the prediction NN is active. It generates the motor commands to the robot based on the time-delayed afferent input received.

C. Afferent Input

The set of neurons representing the discretization of a sensor/hypothesis is activated according to a normalised Gaussian curve. The mean of the Gaussian curve, μ, is at the actual sensor/hypothesis value. See Eq. (1), where χ is the value represented by the neuron and σ the standard deviation of the Gaussian curve used. The σ used here was proportional to the separation in represented value of the neurons, $\sigma = r/2p$ for a value range r and set-size p. See also the Gaussian activation curves visualized in Fig. 1.

$$v = f(\chi, \mu, \sigma) = exp\left(-\frac{(\chi-\mu)^2}{2\sigma^2}\right), \quad where\ v\ is\ a_j,\ a_j^{hyp}\ or\ e_i. \tag{1}$$

D. Efferent Input/Output

In the NN used for learning, the motor commands performed by the teacher is received as efferent input. The same normalised Gaussian activation curve as for the afferent input is used, see Eq. (1). In the NN used for prediction the neural activation in the afferent input layers is propagated to the efferent output, where the robot's own motor prediction and output is generated. Here the ith neuron is denoted as e_i^{pred}. As can be seen in Eq. (2), the activation of each of the n neurons in the efferent output layer (e_i^{pred}), is simply the sum of the m products of the activation of a given afferent input neuron (a_j or a_j^{hyp}) with its respective synapse weight ($w_{i,j}$). The value represented by the neuron with the maximum activation is taken as the respective joint velocity command.

$$e_i^{pred} = \sum_{j=1}^{m} w_{i,j} u, \quad where\ u\ is\ a_j\ or\ a_j^{hyp}. \tag{2}$$

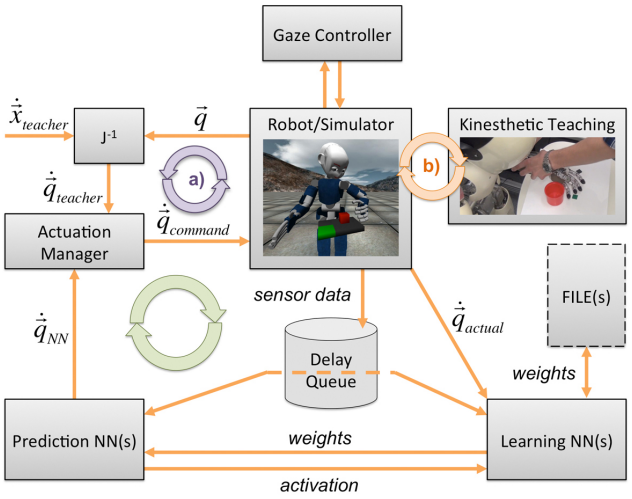


Fig. 2. Overall architecture and embedding in the sensorimotor coordination loop. Large green circular arrow indicates the robot actuation loop, small circular arrows indicate the two possible teacher actuation loops: a) teaching a physical or simulated robot through a user interface generating Cartesian velocities (e.g. a joystick), and b) performing kinesthetic teaching on a physical robot (physically moving the robot). J indicates the Jacobian.

E. Predictive Hebbian Association

In the NN for learning no activation is propagated and a prediction-based learning rule is applied across all synapses. See Eq. (3). The change of a given synapse weight each iteration ($\Delta w_{i,j}$) is proportional to the activation of the respective afferent input neuron (a_j or a_j^{hyp}) and the difference in actual and predicted activation of the efferent command neuron. The rule is a derivative of the Hebbian [15] learning of our previous work [2], but uses the predictions made in parallel to attempt to limit the synapse weights. Similar Hebbian learning rules that correlate an error term with presynaptic activity have been explored in NN architectures with direct biological interpretations [16], [17].

$$\Delta w_{i,j} = \eta y \left(e_i - e_i^{pred} \right), \quad where\ y\ is\ a_j\ or\ a_j^{hyp}. \tag{3}$$

F. High-Level Hypothesis Labels

High-level input can be provided to the NNs through the hypothesis labels. An example could be a specific text label describing the task, like "pick up cup", that the teacher gives to the robot during learning. Each hypothesis input is a binary value (e.g. the existence, or not, of a given label), but the representation is spread over a set of neurons. A pattern generator is used to give a time-variable pattern to the neural activation. A sinusoidal pattern was selected for the implementation presented here, as seen in Eq. (4). Here τ is the desired period of the pattern (here 60 seconds). In the learning NN the activation of a hypothesis will thus continuously activate different neurons, the activation of which are associated with the current motor commands. The activation of a hypothesis during robot actuation can then be used to impose a given trajectory of motor commands.

$$\mu^{hyp} = \lambda cos(2\pi t/\tau),\ where: \quad \lambda = \begin{cases} 1 & with\ hypothesis, \\ 0 & otherwise. \end{cases} \tag{4}$$

G. Embedding in a Developmental Robot Platform

The overall architecture for embedding the NN in the real-time control loop of a developmental robot an be seen in Fig. 2. The YARP [18] communication protocol was used. An actuation manager directly controls the joint velocities of the left arm of the iCub humanoid robot ($\dot{\vec{q}}_{command}$), based on either teacher input ($\dot{\vec{q}}_{teacher}$) or the predictions of the NN ($\dot{\vec{q}}_{NN}$). A separate NN performs the learning simultaneously, and updates the weights used for prediction periodically. Both NNs receive input from the same set of sensors, from the present time and from given times in the past. This is achieved through a delay queue. The gaze of the iCub is made to follow the left hand independently. Examples of learned NN weights can be seen in Fig. 9 for joint encoders, and Fig. 15 for visual sensors.

IV. Simulated Robot Experiment

A. Introduction

The experiment was performed in the iCub simulator [19]. The task setup consisted of a small planar table in front of the robot, and a small red cube (40x40x40 mm). See Fig. 3 for the table used, Fig. 4 for the general setup. The goal of each trial was to push the red cube to one of the two possible target locations on the far side of the table, shown in green. The robot was to learn to take over control from the teacher after a preset time into a demonstration (2 or 4 seconds) and complete the task. The main performance measure was the percentage of successful attempts. The hand of the robot started in the same initial position for each trial, while the red cube was initialized according to a random set of positions within a predefined area. Two different square areas were used for the random distribution, with the size (denoted as d) being 40 or 20 mm. In Fig. 3 dashed black lines indicate the borders for the center of the cube, and the grey areas the outer edges.

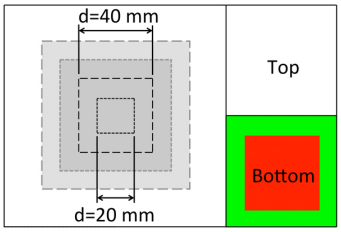


Fig. 3. The "object-pushing" task.

B. Method

1) Teacher Interaction: The teacher used a 6 Degree Of Freedom (DOF) 3DConnexion SpaceNavigator input device to control the Cartesian velocities of the left hand ($\dot{\vec{x}}_{teacher}$ in Fig. 2). The iKinGazeCtrl gaze controller [20] was used to make the robot track the position of the left hand of the robot with the eyes and head. The teacher was provided with

a graphical representation of the average Root Mean Square Error (RMSE) over the 6 predicted joint velocities, see Fig. 4. This provided a tool for the teacher to assess how well the robot was able to predict the current movement performed.

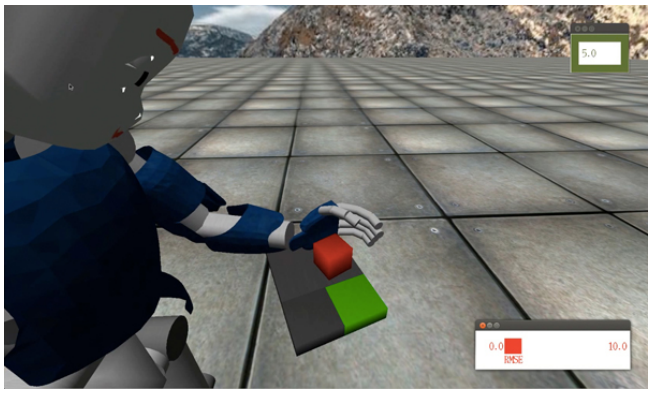


Fig. 4. View given to teacher during testing. Timer in top-right corner, visualisation of average RMSE over joint velocities in bottom-right corner.

2) Sensor Input: Sensor data delayed by 0, 1.5 and 3 seconds was fed to the NNs. The sensors used were: a) The 6 joint encoders of the left arm, b) the 2 joint encoders of the neck, and c) 54 vision sensors (18 red, 18 green, and 18 blue pixel counters, see Fig. 16). See Table I for the assumed resolution (the number of neurons used), min and max values. Each hypothesis was represented by 35 neurons.

TABLE I
SPECIFICATIONS OF SENSORS USED IN EXPERIMENT.

		Resol.	Min.	Max.	Units
Arm	Shoulder pitch	35	-100	15	*degrees*
	Shoulder roll	35	-5	166	
	Elbow	35	10	111	
	Wrist pronosupination	35	-95	95	
	Wrist pitch	35	-95	5	
	Wrist yaw	35	-25	45	
Neck	Pitch	35	-65	-20	*degrees*
	Yaw	35	-5	40	
Vision	24 small pixel counters	20	120	1600	*pixels*
	30 large pixel counters	20	400	3200	

3) Robot Actuators: The actuators used were 6 of the left arm motors (joint velocities commanded), see Table II. Shoulder yaw was kept at 7 degrees. The robot actuation was stopped when the average maximum neural activation across the 6 joint velocities fell below 0.1. The system was tested on an 8-core Dell i7-2600 @ 3.4 GHz. Both the learning and prediction were split over separate NN modules, by running two instances with half the joint velocities for each. The learning achieved 10-15 Hz and the prediction 30-35 Hz.

4) Experiment Procedure: The main experiment conditions can be seen in Table III. A total of about 10 hours of testing was performed, with one of the authors as the teacher. For training sessions, the teacher could switch between own and robot actuation with a button. 40 training attempts were performed. The robot only learned during periods of teacher actuation, and only from successful attempts. For testing sessions the teacher started actuating, and the robot automatically took over after a preset time had passed (2 or 4 seconds). 40 attempts were used for measuring performance for each experiment condition. A separate hypothesis for each target location was given on conditions 3, 4, 7 and 8.

TABLE II
SPECIFICATIONS OF ACTUATORS USED IN EXPERIMENT (IN DEG/S).

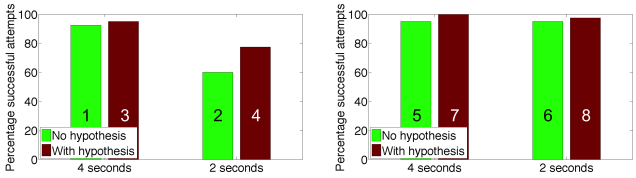
		Resolution	Min.	Max.
Arm velocity	Shoulder pitch	45	-15	15
	Shoulder roll	45	-15	15
	Elbow	45	-15	15
	Wrist pronosupination	33	-15	15
	Wrist pitch	33	-15	15
	Wrist yaw	33	-20	20

TABLE III
MAIN CONDITIONS FOR EXPERIMENT.

		d=40 mm	d=20 mm
No hypothesis	4 sec.	Condition 1	Condition 5
	2 sec.	Condition 2	Condition 6
With hypothesis	4 sec.	Condition 3	Condition 7
	2 sec.	Condition 4	Condition 8

C. Main Results

1) Percentage of Successfully Completed Attempts: The percentage of successfully completed attempts for the main experiment conditions can be seen in Fig. 5. There was a high success rate for most conditions, but the performance dropped sharply when both a high dispersion of the initial locations of the cube (d=40 mm) and a short amount of time before beginning actuation (2 seconds) were used. Providing hypotheses on the given target locations improved performance by 29.2% for this situation, and had a positive effect in general.



(a) d=40 mm. (b) d=20 mm.

Fig. 5. Percentage of successfully completed attempts for the main experiment conditions. With or without hypothesis about task to perform, and beginning actuation after 4 or 2 seconds. All results with 40 attempts for training. Condition number indicated on each column.

2) Trajectories Followed: The differences in the joint space trajectories followed for the two tasks are shown in Fig. 6. It can be seen that the trajectories for the two tasks are quite similar in shoulder roll, while for other joints they differ partially (e.g. shoulder pitch and elbow) or completely (wrist pitch). The different start locations of the red cube

also required quite different trajectories in some joints (e.g. wrist yaw for the top target). In Fig. 7 the corresponding neck angles are shown. The trajectories can mainly be distinguished in the neck pitch joint.

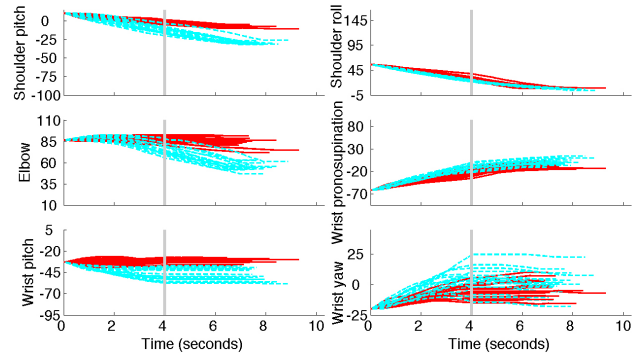


Fig. 6. Arm joint angle trajectories for actuation from 4 seconds (no hypothesis given and d=40 mm; condition 1), for all successful attempts. Both bottom target (solid red lines) and top target (dashed cyan lines). Grey vertical line indicates start of robot actuation.

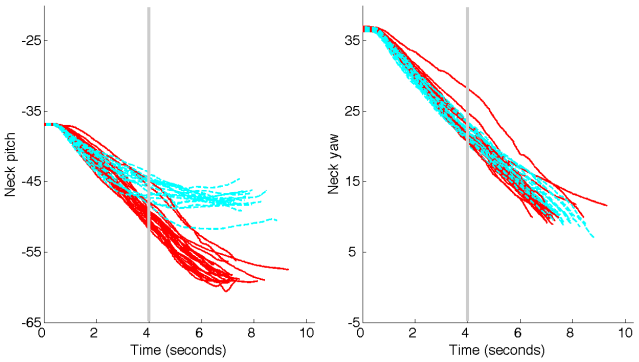
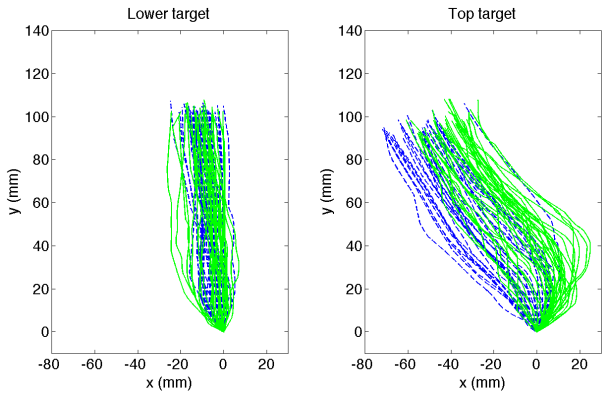


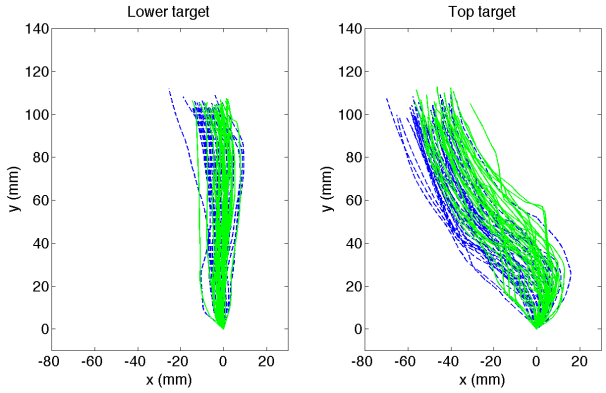
Fig. 7. Neck pitch and yaw angle trajectories for condition 1 (see Fig. 6).

Fig. 8 shows the Cartesian trajectories of the hand in the horizontal plane. It can be seen that many of the trajectories for the two targets cross, making the task of distinguishing one from the other more difficult. The large dispersion of final positions follows from one of the characteristics of the tasks performed, the ability to push the cube with any part of the hand/wrist/fingers.

3) Development of Neural Network Weights: Limiting the synapse weights in Hebbian learning is often addressed through Oja's rule [21] or similar approaches for "forgetting". For a NN that is directly embedded in the real-time loop of a robot agent, the weights will already be somewhat limited. That is, the robot will only be interacting with the world for a given period of time and at a given rate (10-20 Hz here). However, some neural mechanism for normalizing the weights is still desirable. The error in the prediction is here used, which means the NN mainly "forgets" (and learns) when the efferent prediction is far from the efferent command. That is, both learning and "forgetting" depend on the performance in the current context. As can be seen in Fig. 11(a) the NN weights increased sharply

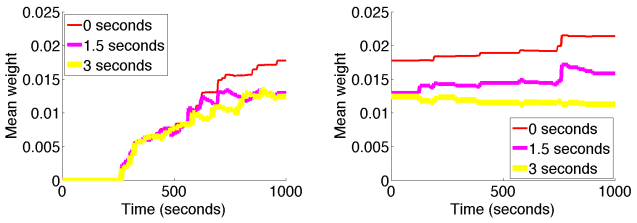


(a) d=40 mm.



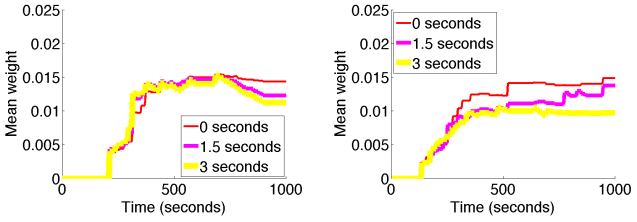
(b) d=20 mm.

Fig. 8. Cartesian trajectories of the hand in the $x-y$ plane (top view) for learning (solid green lines) and actuation (dashed blue lines) trials.



(a) Original 40 training attempts. (b) Additional 40 training attempts.

Fig. 11. Development of the mean of the NN weights for no hypothesis given and d=40 mm (conditions 1 and 2). Over the original 40 attempts and after 40 additional attempts. Mean shown individually for 3 delays used.



(a) With d=20 mm. (b) With hypothesis given.

Fig. 12. Effect on the development of the mean of the NN weights when: a) d=20 mm (conditions 5 and 6), and b) when hypothesis is given (conditions 3 and 4). Mean shown individually for 3 delays used.

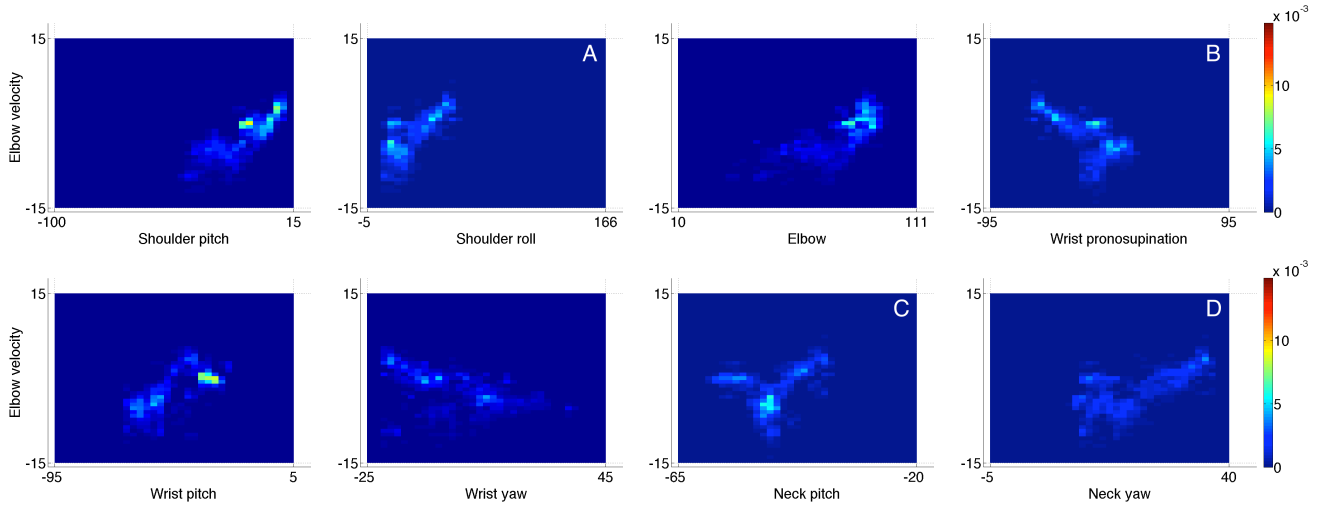


Fig. 9. Example learned neural network weights from arm and neck encoders to elbow joint velocity, at delay of 0 seconds.

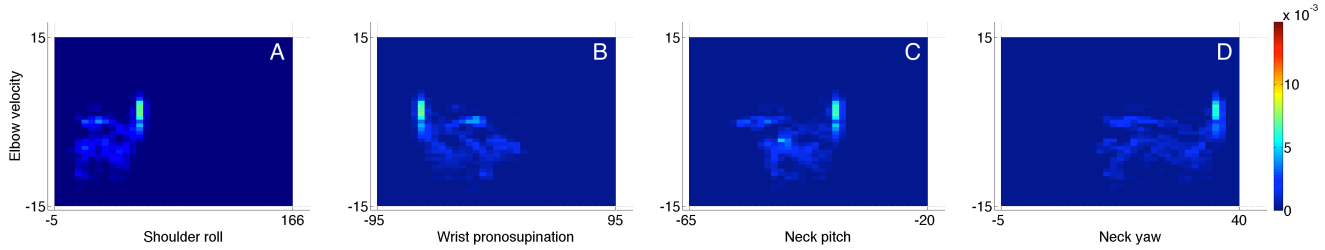


Fig. 10. Example learned neural network weights from arm and neck encoders to elbow joint velocity, at delay of 3 seconds.

at first, but had a gradual reduction in slope towards the end of the 40 training attempts. There was little overall increase in the mean weights when running an additional 40 training attempts after the experiment, as can be seen in Fig. 11(b). However, there was a sudden increase after about 750 seconds that likely stems from variability in the teacher demonstrations. Similar effects were also seen in the other conditions. Overall it seems the training used was sufficient to stabilise the weights for the tasks performed. Fig. 12(a) shows that a less variable initial distribution of positions (d=20 mm) made the tasks easier to learn, with the mean weights quickly reaching a high level. Fig. 12(b) shows the development with d=40 mm and hypothesis given.

4) Final Neural Network Weights: An example of the final NN weights learned can be seen in Fig. 9, for a delay of 0 seconds. For most sensors there are multimodal distributions of weights to the possible motor actions, suggesting non-trivial relationships between sensing and actuation. Fig. 10 show four matching results for a delay of 3 seconds. It can be seen that the weights at the two delays have some observable differences, even though only 3 seconds separate the two sets of weights. The NN architecture presented can take advantage of such additional information by basing the actuation on the recent past, not only the current sensor input.

D. Sensitivity Analysis

The effect of three changes to the sensorimotor dynamics were explored. All three changes were applied to experiment condition 1 (actuation from 4 seconds, no hypothesis given and d=40mm).

1) Learning an Additional Task: The first effect explored was the addition of a different task, touching one of two static spheres, hereby denoted as the "two-button" task. See Fig. 13. One of the spheres were green, the other red, with the order assigned randomly. The goal of the task was to push with the palm against the green sphere. The NN weights obtained during training on condition 1 was first loaded. The NNs were then trained on 40 attempts on the "two-button" task, and was tested to confirm 100% successful execution after 4 and 2 seconds. The robot was then tested on the original task, and was able to complete 82.1% of the attempts. This compared to the 92.5% before learning the additional task, a reduction of 11.2%.

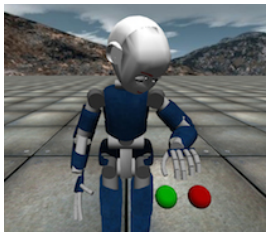
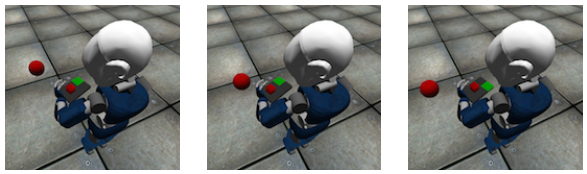


Fig. 13. "Two-button" task.

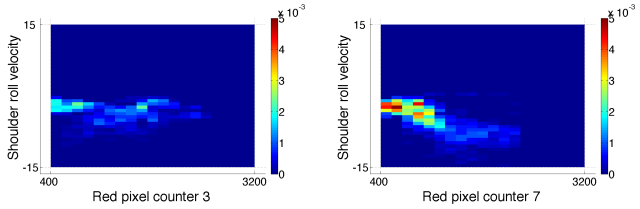
2) Random Visual Effect: Condition 1 was rerun with a random visual effect, a red sphere placed in a randomised location within a 400 mm square on the floor. See Fig. 14 for three examples. The visual impact was approximately the same as the smaller red cube on the table, but occurred mainly in the top half of the robot's retina. The robot was trained with 40 attempts as before. During testing the same

success rate as without the random effect was achieved, that is 92.5%. The corresponding NN weights for the areas of the retina affected by the random sphere were also lower and more even that those affected by the red cube, see the example in Fig. 15. The sensors with the highest respective weights are shown, sensors 3 and 7 (see Fig. 16(a)).



(a) (b) (c)

Fig. 14. Including a red sphere in a random location in front of the robot.



(a) Sensor 3. (b) Sensor 7.

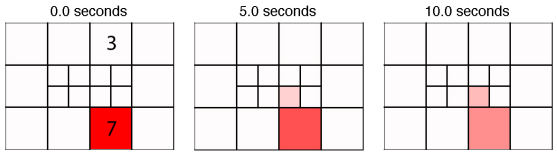
Fig. 15. Example neural network weights from red pixel counters 3 and 7 to shoulder roll velocity. See Fig. 16(a) for a visualization of the sensors concerned.

3) Using Vision Sensing Only: Condition 1 was rerun with only vision input, with 40 attempts for training as before. See Fig. 16 for example visualisations of the visual information received, and Fig. 17 for the corresponding situations. The robot was able to complete 90% of the tasks after 4 seconds without hypothesis given (condition 1). The reduction in sensors led to faster execution of the NNs, up to 13-16 Hz for the learning and 37-40 Hz for the prediction.

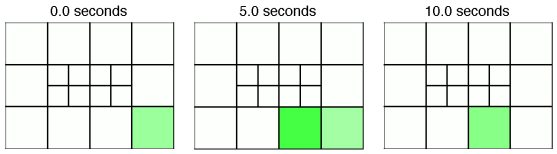
E. Discussion

The tasks used here are quite simple and do not require great precision. However, they have a high degree of overlap and are performed in a small part of the workspace. Tasks involving a greater sensor data diversity may prove easier to learn, at least when the additional data is correlated to the task. Humans are also good at taking advantage of the richness of the context to drive action. Remembering the pin code for a bank card is often much easier in the right context, i.e. with your fingers on the cash machine keypad. In fact, both motor memory formation and decay has been found to be strongly context dependent [22].

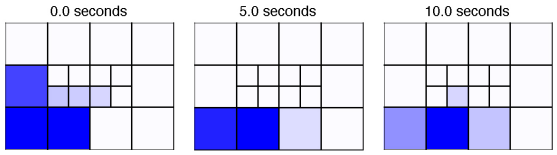
The learning could also have been "hard-coded" to improve performance. First, a separate set of NN weights could have been used for each target, or even for each approximate start location. That is, each movement "class" could have been modelled and then represented in code by a symbol, and only activated when certain predefined conditions were satisfied. Second, estimating the 3D pose of the specific cube of interest with more elaborate vision algorithms is



(a) Red pixel counts.

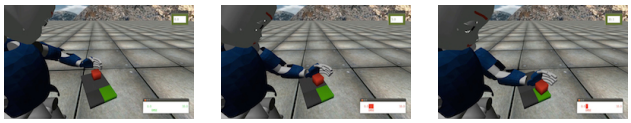


(b) Green pixel counts.



(c) Blue pixel counts.

Fig. 16. Visualizations of visual input at three times for one attempt.



(a) 0.0 seconds. (b) 5.0 seconds. (c) 10.1 seconds.

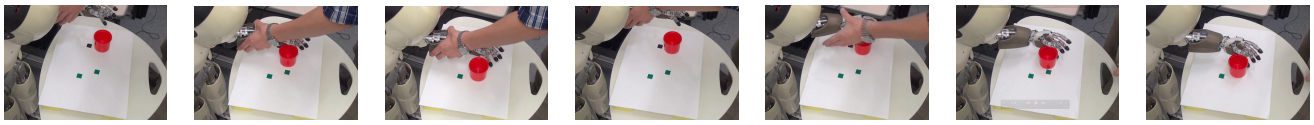
Fig. 17. Actual situations corresponding with Fig. 16.

certainly feasible, but would require a second mechanism for estimating from the context when the 3D pose of the red cube is of interest, and when not. Reducing the need for such gate-selection mechanisms [9] may help circumvent deeper issues like the symbol grounding problem (see [23]).

The current two-NN architecture could be replaced with one NN. This would require neurons in the efferent layer capable of: 1) receiving propagated activation from the afferent layers, 2) activating based on the summed activation received and generating a prediction, 3) being sensitive also to a given value of the actual efferent command, 4) comparing the "predicted" activation and the "actual", and 5) modifying the incoming synapse weights based on this comparison.

V. Implementation on Physical Robot

Two simple tests of the ability to learn and actuate on the physical iCub robot were performed. Kinesthetic teaching was used, where the teacher directly moved the joints of the robot. See teacher actuation loop b) in Fig. 2. For the first test, the task was to move a red cup from one of two points on a table to one of two targets to the right. See Fig. 18. The input to the NN was the 7 left arm encoders and the yaw neck encoder. The head tracked the red cup using a colour segmentation algorithm, see [24]. A different hypothesis was given to each of the four possible tasks and used during actuation. Only one kinesthetic demonstration was given for each task. For the second test, no hypothesis was given, and



(a) (b) (c) (d) (e) (f) (g)

Fig. 18. First learning on the real iCub platform. a) to c): complete user demonstration; d) and e): user starting task; f) and g): robot completing task.

(a) (b) (c) (d) (e) (f) (g)

Fig. 19. Demonstrating simple sensorimotor coordination. a) to c): complete user demonstration; d) user starting task; e) to g): robot following cup.

the robot was shown one task, moving the cup from one point to another. The teacher started the test, then grabbed the cup and slowly moved it to the final point. As can be seen in Fig. 19, the robot followed the cup with the hand, showing a simple (task-limited) sensorimotor coordination.

VI. Conclusions

We have presented a NN architecture that associates time-delayed sensor input with motor commands in a developmental robot platform. The two NNs operate at 10-20 Hz while embedded in the real-time sensorimotor coordination loop. The input from 62 sensors is used directly, without first being reduced to a predefined low-dimensional "state". Actuating a 6 DOF kinematic chain, the NNs are able to learn some simple object interaction tasks, even with noisy effects in the object location and the visual scene. The predictive Hebbian associative rule effectively limits the NN weights based on the error in the predictions made. In addition, the NNs showed promise in predicting the intent of the teacher during movements, and to smoothly complete tasks begun by the teacher. We believe this is an important capability when approaching how children learn and interact. Future work should explore the neural network storage capacity, especially the role of the typically non-uniform and highly "embodied" sensor space (i.e. dependent on the motor system) of a humanoid robot like the iCub.

References

[1] Bernstein, N., "The Coordination and Regulation of Movements." New York: Pergamon Press, 1967.

[2] M.F. Stoelen *et al.*, "Online Learning of Sensorimotor Interactions using a Neural Network with Time-Delayed Inputs," *IEEE Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, San Diego, USA, 2012.

[3] A.J. Ijspeert, J. Nakanishi and S. Schaal, Learning Attractor Landscapes for Learning Motor Primitives, *Advances in Neural Information Processing Systems*, pp. 1547-1554, 2003.

[4] P. Pastor *et al.*, "Online movement adaptation based on previous sensor experiences," *in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 365-371.

[5] S. Schaal and C.G. Atkeson, Constructive incremental learning from only local information, *Neural Computation*, vol. 10, pp. 2047-2084, 1998.

[6] D.H. Grollman and O.C. Jenkins, "Incremental Learning of Subtasks from Unsegmented Demonstration," *in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 261-266.

[7] T. Waegeman *et al.*, "Modular reservoir computing networks for imitation learning of multiple robot behaviors", *in Proceedings of IEEE Inter. Symp. on Comput. Intelligence in Robotics and Automation*, 2009, pp. 27-32.

[8] J. Tani and M. Ito, Self-Organization of Behavioral Primitives as Multiple Attractor Dynamics: A Robot Experiment, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 3, no. 4, pp. 481-488, 2003.

[9] Y. Yamashita and J. Tani, Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment, *PLoS Computational Biology*, vol. 4, no. 11, 2008.

[10] E. Demeester *et al.*, User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving, *Journal of Autonomous Robots*, vol. 24, pp. 193-211, 2008.

[11] B. Akgun, D. Tunaoglu and E. Sahin, "Action Recognition Through an Action Generation Mechanism", *in Proceedings of Tenth International Conference on Epigenetic Robotics*, Orenas, Sweden, 2010.

[12] S. Calinon and A. Billard, What is the teacher's role in robot programming by demonstration? Toward benchmarks for improved learning, *Interaction Studies*, vol. 8, no. 3, pp. 441-464, 2007.

[13] H. Burgsteiner, Imitation learning with spiking neural networks and real-world devices, *Engineering Applications of Artificial Intelligence*, vol. 19, no. 7, pp. 741-752, 2006.

[14] A. White, J. Modayil and R.S. Sutton, "Scaling Life-long Off-policy Learning," *IEEE Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, San Diego, USA, 2012.

[15] D.O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*, Wiley, New York, USA; 1949.

[16] P. R. Montague, P. Dayan and T.J. Sejnowski, A framework for mesencephalic dopamine systems based on predictive Hebbian learning, *The Journal of neuroscience*, vol. 76, no. 5, pp. 1936-1947, 1996.

[17] D. MacNeil and C. Eliasmith, Fine-tuning and the stability of recurrent neural networks, *PloS one*, vol. 6, no. 9, e22885, 2011.

[18] G. Metta, P. Fitzpatrick and L. Natale, YARP: yet another robot platform, *Int. J. Adv. Robot. Sys.*, vol. 3, pp. 43-48, 2006.

[19] V. Tikhanoff *et al.*, "An Open-Source Simulator for Cognitive Robotics Research: the Prototype of the iCub Humanoid Robot Simulator," *in Proceedings of IEEE Workshop on Performance Metrics for Intelligent Systems Workshop (PerMIS'08)*, eds R. Madhavan and E. R. Messina (Washington, DC), 2008.

[20] U. Pattacini, Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub, *Ph.D. dissertation*, RBCS, Istituto Italiano di Tecnologia, Genova, 2010.

[21] E. Oja, A simplified neuron model as a principal component analyzer, *Journal of Mathematical Biology*, 15:267-2735, 1982.

[22] J.N. Ingram, J.R. Flanagan and D.M. Wolpert, Context-Dependent Decay of Motor Memories during Skill Acquisition, *Current Biology*, http://dx.doi.org/10.1016/j.cub.2013.04.079, 2013.

[23] S. Harnad, The Symbol Grounding Problem, *Physica D: Nonlinear Phenomena*, 42:335-346, 1990.

[24] A.F. Morse *et al.*, Epigenetic Robotics Architecture (ERA), *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 325-339, 2010.