A Deep Reinforcement Learning Based Model Supporting Object Familiarization

Maximilian Panzner
Semantic Computing Group,
CITEC, Bielefeld University
mpanzner@cit-ec.uni-bielefeld.de
`https://cit-ec.de/`

Philipp Cimiano
Semantic Computing Group,
CITEC, Bielefeld University
cimiano@cit-ec.uni-bielefeld.de
`https://cit-ec.de/`

***Abstract*—An important ability of cognitive systems is the ability to familiarize themselves with the properties of objects and their environment as well as to develop an understanding of the consequences of their own actions on physical objects. Developing developmental approaches that allow cognitive systems to familiarize with objects in this sense via guided self-exploration is an important challenge within the field of developmental robotics. In this paper we present a novel approach that allows cognitive systems to familiarize themselves with the properties of objects and the effects of their actions on them in a self-exploration fashion. Our approach is inspired by developmental studies that hypothesize that infants have a propensity to systematically explore the connection between own actions and their perceptual consequences in order to support inter-modal calibration of their bodies. We propose a reinforcement-based approach operating in a continuous state space in which the function predicting cumulated future rewards is learned via a deep Q-network. We investigate the impact of the structure of rewards, the impact of different regularization approaches as well as the impact of different exploration strategies.**

I. Introduction

In many tasks, intelligent and cognitive systems such as robots need to familiarize themselves with the physical properties of a certain environment as well as with the consequences of their own actions on this environment. A challenging task for artificial systems in particular is to familiarize themselves with the properties of objects, learning in particular how to manipulate them in a meaningful way.

Our work is related to cognitive robotics and developmental robotics in that we are interested in developing models that allow a system to self-familiarize with a non-trivial articulated object by exploring a huge continuous space with discrete action possibilities guided by the desire to induce change in the object. Our model is developmentally inspired in the sense that a system learns action categories by performing actions on an object and observing the consequences, relying on intrinsic motivation to induce change. In fact, some authors in the field of developmental psychology have argued that young infants have a propensity to systematically explore the connection between their own actions and the perceptual consequences in order to support inter-modal calibration of their bodies [1].

We realize this propensity to systematically explore the connection between own actions and their (perceptual) consequences through a bias that leads learning systems to attempt to induce change to maximize encounters with objects in which something ‘happens’, thus maximizing the situations in which the system can learn something about the effects of its own actions on the environment. We incorporate this bias for inducing change via a reward function that rewards a learning system for inducing movement in an object that the system is familiarizing with.

We implement the system as a deep reinforcement learning model relying on a neural network to compute the action-value function in terms of expected discounted future reward. As a reward is needed to guide the search for an interesting policy, we investigate the structure of rewards that are suited to guide the effective self-exploration of a system towards understanding the effects of their own actions on an object having the sole goal of inducing movement on the object.

Our contributions are as follows:

- We present a new deep learning and reinforcement-based model that allows a system to familiarize itself with the consequences of their actions on a given object in a self-exploration fashion. The self-explorative behavior is guided by a bias to induce change to maximize the chance to encounter interesting learning situations.
- We investigate different reward schemes that capitalize on this bias and give reward for keeping objects moving and quantify their effectiveness to guide a system to induce change on the object that it tries to familiarize with.
- We show the effect of different exploration strategies on the task, showing that a Boltzmann exploration scheme can be beneficial over ϵ-greedy exploration in a task with many degrees of freedom.

The paper is structured as follows: in the next Section II we describe the scenario we tackle, in Section III we describe the reinforcement-based model we use. In Section IV we show how the action-value function can be learned using a deep learning architecture. We present experimental results in Section V and evaluate the impact of different design choices. We describe related work in Section VI and conclude in Section VII.

II. Scenario

In this paper we consider a task where a robot tries to explore meaningful ways to manipulate objects lying on a

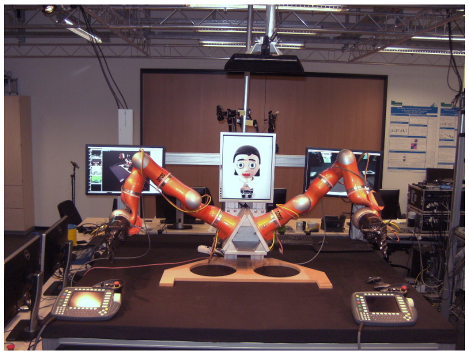

Fig. 1. The FAMULA platform: A bimanual robotic platform supporting the investigation and development of approaches by which a cognitive system can familiarize itself with new objects and concepts through the interplay of manual action and language (https://www.cit-ec.de/en/deep-familiarization-and-learning).

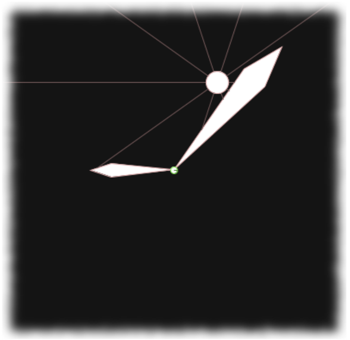

Fig. 2. A simulated clock-like object consisting of two hands rotating around a centre. A simulated fingertip can move the object and observes its environment through 10 star-shaped distance scans to other objects or the boundary of the operation space.

table in front of it. This research is embedded in the FAMULA [2] project, which is concerned with developing approaches by which a bimanual robotic platform can acquaint itself with the properties and affordances of objects by (guided) autonomous self exploration. The platform combines rich bimanual actuation with anthropomorphic hands allowing for dexterous object manipulation (Figure 1). Considering that having to monitor the robot personally during the long-running experiments would have seriously limited the amount of experiments that could have been carried out, the experiments were performed in simulation rather than on the physical platform. However, we assume that findings from experiments in simulation will prove qualitatively comparable to experiments on the platform. In our simulated environment, the actuator is modeled after the tip of a finger of a robotic hand sliding over the table in front of the robot. To evaluate the exploration driven object familiarization, we placed a complex object with rich physical dynamics in the center of the simulated table. The object was modeled after a real toy clock. The robot can explore its simulated environment (Figure 2) by taking physical action and observing the results of its actions in terms of change in the environment and a sparse reward signal indicating how well the robot performed with respect to a measure of object-familiarization. The simulated table provides rich and accurate physics with dynamics that are not only reactive in a sense that changes in the environment only happen in response to actions the robot takes, but reactions can also sustain for a longer time due to, e.g., the interplay between inertia and friction in moving objects. The robot can not observe the full state of the simulation at once, but instead has to infer the state of the simulation from partial observations in terms of a two dimensional circular distance scan to objects in its current vicinity as described in section II-B.

The simulated clock has one large and one small hand which both are freely movable on a circular path around the clock center. A key challenge for the robot in learning the motion dynamics of the clock lies in the special non-euclidean way the clock hands move. To build a forward model which projects the current state to future states of the moving clock hands, the system has to learn to map the euclidean coordinate system it operates in into the polar coordinate system in which the clock hands move. The simulated clock has some more interesting properties for the robot to discover. One side of a clock hand is fixed to the clock center and can not be moved, while the other side is freely movable. Applying a force to the outer side of a clock hand makes it easier to induce movement than applying a force somewhere in the middle of the clock hand due to static friction and the law of the lever.

A. *Actions*

To interact with its environment the robot can take one of 6 actions including actions that accelerate the simulated fingertip in one of 5 equally partitioned directions $(0°, 72°, 144°, 216°, 288°)$ and also one action which does nothing and can be used by the robot to fine-control the movement speed of the fingertip or the force applied to another object.

B. *Observations*

The robot observes its environment through 10 circular distance scans (Figure 2) across the vicinity of the robot. The distance scans have been implemented as raycasts emanating from the robots center position measuring the distance to the first object they reach. This observation scheme is consistent with the state representation of the robotic platform which has an internal model of object positions and extents and can thus steer the fingertip even when the hand blocks the sight of the visual system. The fingertips are also equipped with force sensors that detect when the robot has contact with another object. Having contact with an object would in this setting correspond to at least one of the distance scans being zero.

III. Model

Model free object familiarization is implemented as a reinforcement learning approach where the robot has no prior knowledge about the environment it operates in or about the extend and shape of its own physical manifestation. Reinforcement learning is a machine learning framework where learning is driven by the desire to maximize some notion of cumulative reward R (Equation 1) by iteratively exploring the state-action space of the learning environment through performing actions and observing changes in the environment as a result of the robot's actions. Q-learning [3] (a form of temporal difference learning [4]) is one of the most popular reinforcement learning algorithms but it quickly becomes infeasible when the discretized state-space becomes large due to the curse of dimensionality [5]. Further, a naive discretization of state and action space can hide information that may be crucial to solving the task. In our model we build on Deep Q-Networks [6], which replace the discrete Q-function with a continuous deep neural network, thus acting as universal function approximator, with Double Q-Learning [7], which improves performance by limiting the overestimation of action-values conventional Q-learning is known for. Deep Q-learning problems consist of three components.

- A scalar **reward function** which gives the system positive or negative feedback $r \in \mathbb{R}$ on how well it performs. Future rewards are discounted by a factor γ per time-step, giving precedence to current over later rewards.
$$R = \sum_t \gamma^t r_t. \tag{1}$$
- A **policy** π which determines the optimal action given the current state-action value assessment.
- An **action-value function** which assesses the value of the current states' partial observation $s \in \mathbb{R}^d$ in terms of expected discounted future reward a robot can maximally achieve when taking action a in current state s and following the policy π afterwards
$$Q_\pi(s,a) = \mathbb{E}[r_t + \gamma r_{t+1} + ... + \gamma^n r_n | s_t = s, a_t = a]. \tag{2}$$

To interact with the environment the robot has the ability to perform actions that have direct or delayed effects on the future state of the environment. Unlike in supervised learning where we have one target label for each training example, the typical setting for reinforcement learning is that we have sparse and time-delayed rewards from which the robot has to learn how to manipulate the environment in a meaningful manner. To elicit meaningful training signals from such an environment with sparse reward signals we have to rely on exploration. As the environment is deterministic, better state-action combinations get better rewards on average over time. One of the biggest challenges in reinforcement learning is to find a good trade-off between exploration of the environment and exploitation of already learned state-action combinations. Insufficient exploration could leave the robot with a suboptimal policy while on the contrary excessive exploration might severely slow down learning or even lead to completely flawed policies. The objective of the robot now is to interact with the simulation by selectively applying actions in a way that maximizes its expected future reward.

IV. Q NETWORK

To estimate the Q function we implemented a 6 layer neural network (Figure 3) similar to the deep q-learning network (DQN) in [6], which estimates the expected discounted future reward for each of the 6 actions. For deterministic policies we can reformulate $Q(s,a)$ (Equation 2) recursively by the so called Bellmann equation:

$$Q_\pi(s,a) = \mathbb{E}[r_t + \gamma Q(s,a) | s_t = s, a_t = a] \tag{3}$$

Instead of having a second parameter to the neural network along with the state vector, the Q network $Q(s,a;\theta)$ is implemented to directly output the value of all 6 actions at once $Q(s,\cdot;\theta)$, where θ are the current parameters of the network. This saves 5 additional forward passes per iteration and is equivalent to calculating the value of each state-action pair separately because in this setting all actions are possible regardless of the current state. The target for deep Q-learning [8] is

$$Y_t = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-). \tag{4}$$

This training target employs a separate target network θ^- which is structurally equivalent to the online network except that its parameters are a periodic copy of the parameters of the online network every τ time-steps. A separate target network is used to decouple the update from the target Q-values used to compute the training loss and thus reducing the probability for the training to fall into a feedback loop between the target and estimated Q-values. Because of the max operation in the target, Q-learning and DQN tend to overestimate action values [7]. We adapt the idea of Double Q-learning [9] and decompose the max operation in the target into action selection and action evaluation. The update for Double DQN uses the online network (θ_t) for action selection and the target network (θ_t^-) for action evaluation under a greedy policy

$$Y_t = R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \theta_t), \theta_t^-). \tag{5}$$

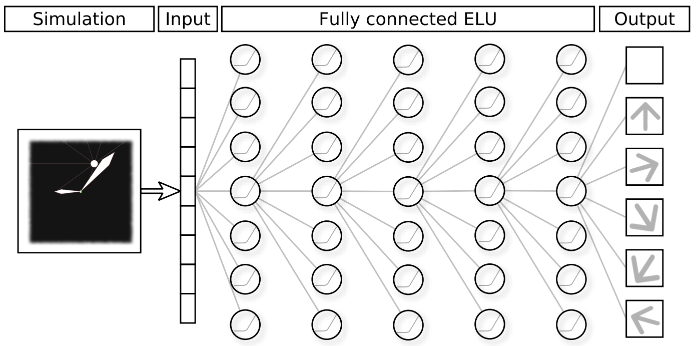

Fig. 3. Layout of the neural network. The robot perceives its environment in terms of a 5-fold circular distance scan yielding a vector of 5 real valued distances to the closest object (including border of operational area) in the corresponding direction. Two successive observations are concatenated into one 10 dimensional vector which is fed as input into the 6-layer network with 96 nodes per layer. Each node in the network is fully connected to all other nodes in the preceding layer. The network uses exponential linear units (ELU) in the first 5 layers and a linear projection in the output

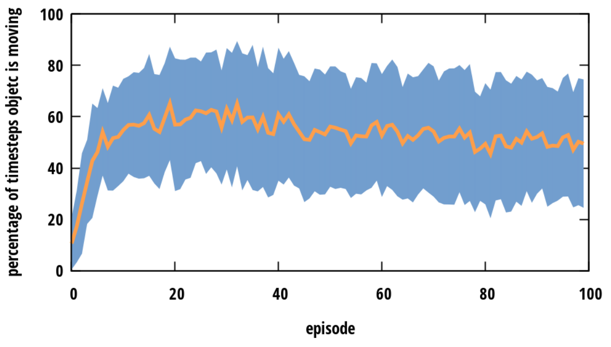

Fig. 4. Evaluation of the systems score after every training episode with goal derived, tutoring and penalty rewards in combination with Boltzmann exploration strategy and L_1 regularization of the network. The shaded area corresponds one standard deviation from the average across 25 evaluation runs

As activation functions for the inner layers we use exponential linear units (ELU) [10]

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{6}$$

instead of the originally proposed ReLU

$$f(x) = \max(0, x) \tag{7}$$

activations because we found that due to the sparse but strong updates in this setting the activations of up to $\frac{1}{4}$ of the network became constantly negative. ReLUs can not recover from constantly negative input because the gradient is zero which leads to vanishing updates. To enable the network to estimate dynamic properties such as the speed of the fingertip or the clock hands we provide not only the current observation but also the observation of the last time-step as history. As input the last two observations are concatenated

$$s_t = o_{t-1} \oplus o_t \tag{8}$$

and normalized to zero mean and unit variance with per-batch normalization [11]. We trained in batches of 32 examples sampled uniformly from buffered experiences which consists of transitions from the last 1500 time-steps. This so called 'experience replay' breaks correlation in strongly correlated data such as time-series in reinforcement learning by storing samples in a fixed-size buffer and training on random (mostly non-successive) subsamples. We use Adam [12] as optimizer, which combines ideas from RMSProp [13] to deal with non-stationary objectives and AdaGrad [14] to deal with sparse gradients as they particularly occur in reinforcement learning.

V. Experiments

To evaluate the ability of the system to familiarize itself with a complex object like the toy clock, we trained the system in various conditions and evaluated the system's performance by measuring the percentage of time-steps within an episode in which the clock pointers where actively or passively moving. Active movement in this case is caused by the robot applying a force to the clock pointers, while passive movement is due to inertia and friction. In other words, the robot's task was to learn to induce as much change as possible in the environment. Policies learned by reinforcement learning are influenced by the reward signal, which helps the robot to structure its perception-action mappings in a goal-directed manner. In other domains like e.g. playing atari video games [6], the reward signal is determined by the game and was specifically designed to give frequent rewards to players to motivate them to keep on playing and improve their gaming performance. The simulated environment in this experiment has no intrinsic reward structure. The robot starts without any prior knowledge about the environment or about itself and its own capabilities, except that in each time-step it can choose one out of 6 different (unspecified) actions. To discover structure in the observations, the robot has to explore the environment by taking actions, observing consequences and assess them in relation to the reward received. In the following experiments we evaluate effects from different reward structures, different exploration strategies and regularization of network weights and biases. Each setting is trained for 100 episodes, where every episode is a complete run of the simulation scenario for 1500 time-steps. To mitigate the impact of randomness (from the random exploration strategies), we ran each experiment 25 times and calculated the mean scores (time-steps the object was moved) per experiment. Network topology and hyperparameters have been optimized separately by a combination of manual optimization and randomized grid search.

TABLE I
Evaluation of method combinations with their score (percentage of time-steps the object was moving) and standard deviations. Bold results represent the same configuration

Method	$score_{max}$	$score_{avg}$	σ_{avg}
Goal derived reward	47	38	4.0
Goal derived + tutoring	69	53	6.8
Goal derived + tutoring + penalty	65	59	3.16
ϵ-greedy exploration	46	39	3.5
Boltzmann exploration	65	59	3.16
No regularization	60	53	7.4
L_1 Regularization	65	59	3.16
Dropout (ρ=.5)	10	5	1.8

Table I lists results for different combinations of reward signals, exploration strategies and network regularizations. The scores are given in two conditions, $score_{max}$ denotes the peak performance when selecting the best model among the 25 runs of the experiment, while $score_{avg}$ denotes the performance averaged over all of the 25 experiment runs. The maximal performance ($score_{max}$) corresponds to a setting where the robot can be trained multiple times in a separate process (off-task) and the best performing model is deployed afterwards. The average score ($score_{avg}$) denotes the expected score when the robot has to be trained on-task, which is probably the most common case, and σ_{avg} denotes the standard deviation over all 25 runs. Bold results in the table correspond to the same configuration displaying the best average performance.

Figure 4 compares the performance of the best model configuration along with the standard deviation across different evaluation runs.

A. Reward Structure

The reward the robot receives from its environment is the primary driving force of the learning process. We investigate different strategies to structure the reward signal. As the simulation permits many degrees of freedom, which renders the state space too large for pure random exploration, we also introduce small tutoring reward signals for the robot to learn promising exploration directions. Tutoring reward ($r = 0.15$) is given with 10% chance when the robot moves towards one of the clock hands. Goal derived reward with magnitude $r = 1$ is given every time-step the robot achieves to move a clock hand. This reward is directly derived from the performance measure and is maximal when the clock hands move at every time-step. Penalty rewards are given when the robot tries to leave its operating area on the table ($r = -1$) or when the robot stalls for more than 10 time-steps ($r = -0.15$). This also helps to pre-structure the exploration space towards the center of the operating area. Table I lists the results of three combinations of reward signals. Goal derived reward alone seems to be insufficient to learn a good policy. Goal derived reward in combination with small tutoring rewards improves the performance of the learned models significantly and achieves the highest maximum score of all models. But as the maximum score is prone to random effects from the exploration strategies, we put more weight on the average score, which is best when all three reward signals are combined.

B. Exploration Strategy

In the domain of atari games [6] the ϵ-greedy exploration strategy proved valuable to find a good trade-off between exploration and exploitation of already learned moves. This policy picks the best action from the current action-value estimates

$$\pi_t = \operatorname*{argmax}_a Q(s_t, a) \tag{9}$$

for the current state s with a probability $1 - \epsilon_t$ or a random action with probability ϵ_t. The exploration factor ϵ is initially set to 1 and linearly decays to 0.01 within the first 10 episodes of training.

Boltzmann exploration [15] is an exploration scheme that takes the relative action values into account. In contrast to ϵ-greedy it is adaptive in a sense that in states with clearly advantageous actions it behaves like a greedy strategy and follows optimal actions, while in states where all action-values are similar the Boltzmann exploration introduces more randomness. Similar action values could arise in states that are universal in a sense that it does not matter much which action to take. Similar values can also arise in poorly explored states where the network is uncertain about which action is the best action to take. Boltzmann exploration is similar to drawing samples from a softmax distribution over the Q-values but takes also a temperature factor T into account which is annealed over the first 10 episodes of training:

$$\pi_t(a) = \frac{\frac{e^{Q(s_t,a)-Q(s_t,a_{max})}}{T_t}}{\sum_a \frac{e^{Q(s_t,a)-Q(s_t,a_{max})}}{T_t}} \tag{10}$$

where $a_{max} = \operatorname{argmax}_a Q(s_t, a)$. With temperatures close to zero, this exploration scheme behaves like Equation 9, while with higher temperature values it selects actions more randomly. As can be seen in Table I, the Boltzmann exploration scheme clearly outperforms the ϵ-greedy exploration strategy in this setting. One observation from visually inspecting the simulation is that ϵ-greedy explores only a limited area of the operation space because actions are contradictory and selecting them randomly cancels their effect in average. Boltzmann exploration on the other hand takes already learned action values into account when randomly selecting actions, which, in combination with tutoring rewards, leads to early movements towards the object and better average evaluation score.

C. Regularization

In this experiment we investigate the impact of different regularization methods on the performance of the system. One observation while training different models was that even when the simulated fingertip was close to a clock hand the Q-values did not differentiate well. The expectation was that when the current state offers a clear opportunity to receive reward, the values of actions bringing the fingertip closer to a clock hand would lead to clear peaks in the action value assessment of the Q-network. But even in opportunistic states the Q-values where close and hence the best action was prone to change during training. To measure the closeness of action values one could adapt the idea of entropy over distributions by first performing a softmax normalization and calculating the entropy as the expectation over the information content

$$H(A) = -\sum_a p(a) \log p(a) dx \tag{11}$$

as a measure of surprise in the action values. We found that applying L_1 regularization improved differentiation of Q values in opportunistic states and also improved the behavior of the robot when visually inspecting the simulation. However, this effect is not well reflected in the performance measure applied in these experiments but could be useful for tasks with other goal definitions. The regularization strength ($\lambda = 0.1$) was optimized by grid search. L_2 regularization was also applied but performed minimally worse than L_1. Another often applied regularization technique in deep neural networks is dropout [16]. Dropout randomly drops units from the network with a certain probability during training and thus simulates the training of many smaller subnetworks, which is similar to the idea of ensemble methods. This technique was proposed to lower overfitting by limiting the co-adaptation between units in the network. Dropout forces the network also to develeop a distributed representation of inputs rather than having parts of the network specialize on specific stimuli, because when randomly dropping units later units cannot rely on input from earlier units in the network. Interestingly, applying dropout with $\rho = 0.5$ in this setting lowered the performance of the robot drastically (see Table I), which suggests that there are parts of the network which tend to specialize on specific (probably location specific) observations.

VI. Related Work

Our work is related to deep learning models that learn to play video games in that we use a deep Q-learning model similar to the one proposed by Mnih et al. [6] with additional double Q-learning (see [7], [9]) and a Boltzmann exploration strategy instead of the originally proposed ϵ-greedy policy. However, it also crucially differs from this line of research in that we are concerned with learning in environments that have less inherent structure than video games that where specifically designed to give frequent reward to keep the players motivated. In contrast to the aforementioned approach, we do not use convolutional layers to learn representations directly from pixels. It has been shown many times that convolutional layers perform fairly well in related tasks so that in this paper we focus on exploration and reward structure and represent observations in a way that is more suited to robot object manipulation where the vision is mostly occluded by the robotic hand.

In developmental robotics, a challenge is to devise models that guide and drive motor and object manipulation learning by intrinsic motivation such as the desire to induce change in the environment or reduce the uncertainty of a forward model that tries to predict future states from perceptions of the current state. Singh et al. [17] investigate intrinsic reward signals in a discrete "playroom" environment where the robot can explore the interactions between different objects such as a light switch, a ball, a bell, movable blocks, etc. Most of the affordances in the environment only apply in interaction between the objects, e.g. the bell rings if the ball is kicked onto it. Intrinsic reward is only generated by unexpected salient events given as error in the prediction of action outcomes. Schmidhuber [18] proposed using curiosity and boredom as confidence-based rewards signals, where curiosity is used as a driving force to explore areas of the action-state space where the predictive model has low confidence.

VII. Conclusion

We have presented an approach allowing cognitive systems to engage in guided self-exploration of object affordances with the goal of understanding the consequences of their own actions on the given object. The self-exploration is guided by the desire to induce change as an intrinsic goal. We have proposed a reinforcement learning based model using deep Q-networks to compute the state-action values as cumulated future reward. We have applied the model to a simulated robot environment consisting of a robotic "fingertip" that can slide over a simulated (table) surface and manipulate the arms of an analog toy clock. We have shown that deep Q-learning is a suitable framework to support robot object familiarization. An interesting follow-up experiment would be to attempt to develop higher level goal-directed action concepts, such setting the clock to a specific time, based on the fundamental object manipulation capabilities acquired through this approach.

Acknowledgment

This research/work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

[1] P. Rochat, "Self-perception and action in infancy," *Experimental Brain Research*, vol. 123, no. 1-2, pp. 102–109, 1998.

[2] "Deep familiarization and learning grounded in cooperative manual action and language." [Online]. Available: https://www.cit-ec.de/de/deep-familiarization-and-learning

[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[4] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988. [Online]. Available: http://dx.doi.org/10.1007/BF00115009

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[7] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *arXiv:1509.06461 [cs]*, no. 2, pp. 1–5, 2015.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[9] H. van Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[10] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *Under review of ICLR2016 ELU*, no. 1997, pp. 1–13, 2015.

[11] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Arxiv*, pp. 1–11, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[12] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," *International Conference on Learning Representations 2015*, pp. 1–15, 2015.

[13] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, p. 2, 2012.

[14] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 2017, vol. 2.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[17] S. Singh, A. Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning," *18th Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 17, no. 2, pp. 1281–1288, 2004.

[18] J. Schmidhuber, "A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers," *Meyer, J.A. and Wilson, S.W. (eds) : From Animals to animats*, vol. 1, pp. 222–227, 1991.