2017 Joint IEEE International Conference on Development and Learning and
Epigenetic Robotics (ICDL-EpiRob)
Lisbon, Portugal, September 18-21, 2017

Curiosity-Driven Exploration Enhances Motor Skills of Continuous Actor-Critic Learner

Muhammad Burhan Hafez, Cornelius Weber, Stefan Wermter
Knowledge Technology, Department of Informatics, University of Hamburg,
Vogt-Koelln-Str. 30, 22527 Hamburg, Germany
Email: {hafez, weber, wermter}@informatik.uni-hamburg.de
http://www.knowledge-technology.info

***Abstract*—Guiding the action selection mechanism of an autonomous agent for learning control behaviors is a crucial issue in reinforcement learning. While classical approaches to reinforcement learning seem to be deeply dependent on external feedback, intrinsically motivated approaches are more natural and follow the principles of infant sensorimotor development. In this work, we investigate the role of incremental learning of predictive models in generating curiosity, an intrinsic motivation, for directing the agent's choice of action and propose a curiosity-driven reinforcement learning algorithm for continuous motor control. Our algorithm builds an internal representation of the state space that handles the computation of curiosity signals using the learned predictive models and extends the Continuous-Actor-Critic-Learning-Automaton to use extrinsic and intrinsic feedback. Evaluation of our algorithm on simple and complex robotic control tasks shows a significant performance gain for the intrinsically motivated goal reaching agent compared to agents that are only motivated by extrinsic rewards.**

Index Terms— Reinforcement Learning; Motor Control; Curiosity-based learning; Predictive Model; CACLA.

I. INTRODUCTION

Helping autonomous agents to learn new motor skills from trial-and-error experience in their environment is the major task of Reinforcement Learning (RL). Learning is achieved when the agent keeps executing actions that maximize long-term reward from its environment, thus employing an optimal policy describing the desired skill. During learning, in deciding on its next action, the agent faces the exploration-versus-exploitation trade-off, a key problem in RL, particularly in complex domains. This is because inadequate exploration is most likely to keep the agent from discovering effective control policies.

To address balancing exploration and exploitation, different approaches adopt different strategies. Some simply use random exploration methods, such as the commonly used ε-greedy and Boltzmann exploration [1]. This includes even a recent breakthrough in RL for playing Atari games [2]. Other approaches use count-based exploration to keep track of the state-action visitation frequency, encouraging the agent to try actions it has a lot of uncertainty about, with exploration bonuses [3] and R-MAX [4],[5] being the most popular. A more recent approach applies Thompson sampling, learning multiple action-value functions via bootstrapping [6].

What is common among these approaches is their strong reliance on external rewards; however, fully autonomous agents often operate in environments where such external rewards do not exist or are sparsely available. Intrinsically motivated RL, therefore, attempts to address this common limitation by endowing the agent with intrinsic drives/rewards, such as fear, hunger or curiosity, which enable it to continue to meaningfully explore its environment and gather useful experience data to learn from [7].

A wide variety of functions have been defined for computing an intrinsic reward, including Bayesian surprise [8], measures based on the agent's learning progress in predicting action outcomes [9], [10], information theoretic measures like information gain [11] and empowerment [12], measures based on novelty of observed states [7],[13],[14] and measures based on the change of the agent's policy value, which is the expected value of the start state [15],[16].

The majority of these functions focus almost entirely on the perceptual characteristics of the observed information. Only the learning progress-based measures have allowed the agent's learning history from previous experience as well as the environmental features to define the agent's curiosity as an intrinsic reward [9], [10]. This is also supported by recent studies from developmental psychology that have found infants' curious exploration to be based on both their own learning history and perceptual variability. These studies show evidence of a Goldilocks effect where seeking just the right level of difficulty drives optimal learning with too much or too little difficulty being disruptive [17],[18].

Curiosity as a drive to maximizing the learning progress of an artificial agent has been successfully applied to various problems, for example, to the acquisition of increasingly complex behaviors in play experiences in simple spaces, which resemble children's developmental patterns [9] and to learning goal-oriented navigation in a 2D maze-like environment with high-dimensional visual input [10]. None of these approaches, however, address curiosity-based exploration in continuous control. Instead, they use discrete action spaces.

Curiosity models, which employ learning progress-based measures, usually use a single model of the environment to assess a prediction error. Such a model cannot give accurate information on whether learning has progressed in particular regions of the agent's sensory space where the interaction with the environment actually happens. Instead, it only informs us on the prediction performance over the entire sensory space, which is difficult to cover and, often, irrelevant to spatially-local behavioral patterns we are interested in modeling. Furthermore, it requires a collection of sufficient training samples that goes beyond the agent's lifetime. Therefore, in this approach, we train an ensemble of local

978-1-5386-3715-9/17/$31.00 ©2017 IEEE

predictive neural networks, as opposed to one monolithic network. The intrinsic reward is composed of two terms: the perception error of the ensemble, and the change of the prediction error of a local model. These separate terms make the intrinsic reward more stable to fluctuations and the agent more likely to continually discover better control policies.

Learning multiple forward and inverse models has also been used for efficient exploration in continuous, redundant control spaces [19],[20]. As opposed to random exploration of the high-dimensional action space with motor babbling, these approaches perform exploration in the low-dimensional task space for efficient learning of inverse kinematics. [20] proposes active self-generation and selection of high-level goals driven by a measure of competence progress to reach these goals and uses RL for goal-directed learning of policy parameters. [19] requires a predefined set of goals and paths along which goals are ordered and tried to reach to detect and resolve inconsistent samples resulting from redundancy and drifts in the inverse estimate. This approach makes no use of RL of control actions. In contrast to these models, we are interested in improving RL with intrinsic motivation to learn good general policies to achieve the desired task.

More recently, Deep Deterministic Policy Gradient (DDPG) has been proposed for learning continuous motor policies [21]. Similar to our work, DDPG is an off-policy actor-critic algorithm that learns a deterministic target policy while behaving according to a stochastic behavior one. The main difference is that the actor in DDPG updates the policy to follow the gradient of the critic's action-value function, but the actor in our algorithm as well as in [22] updates its policy towards an action that was found to be better than the current approximation of the optimal action. Using gradient ascent on the critic's value for adjusting the policy as in DDPG is prone to early divergence since the gradient of a not-fully-trained critic will not always be accurate, as found in [22] when testing on simple control tasks.

In this paper, we propose a novel curiosity-driven RL algorithm based on an actor-critic model. Our algorithm incrementally builds a network of local forward models that handles the computation of the agent's learning progress-based intrinsic reward. This reward is then used to shape the action selection and direct the agent at potentially informative states and actions that improve the prediction error of its environment dynamics in a continuous state-and-action space.

The paper is organized as follows: Section II gives the necessary background on the Continuous-Actor-Critic-Learning-Automaton algorithm. Our proposed algorithm is then described in Section III. Empirical evaluation and results are shown in Section IV. We conclude in Section V by summarizing the main results and providing directions for future work.

II. BACKGROUND

We consider a standard RL setup with an infinite-horizon discounted Markov Decision Process (MDP), where an agent decides on an action and observes a new state and reward. An MDP is defined by a tuple (S, A, T, R, γ), consisting of a set of states S, a set of actions A, state transition distribution $T : S \times A \times S \rightarrow [0, 1]$, reward function $R : S \times A \times S \rightarrow \mathbb{R}$ and discount factor γ, $0 \leq \gamma < 1$. We aim to find a behavioral policy $\pi : S \times A \rightarrow [0, 1]$ that maximizes the total discounted reward $\sum_{t=0}^{\infty} \gamma^t r_t$. The state value function $V^{\pi}(s)$, given a policy π, is defined over all states and indicates the expected total discounted reward when executing policy π from state s. The optimal value function corresponding to the optimal policy π^*: $\pi^* = \arg\max_{\pi} V^{\pi}(s)$ is given by:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')\big(R(s,a,s') + \gamma V^*(s')\big) \qquad (1)$$

which is the Bellman optimality equation for V^* [1].

We are concerned with online RL, where T is unknown and the agent receives a sample transition (s_t, a_t, s_{t+1}, r_t) at each time step and uses it to update an estimate of the state value function via Temporal Difference (TD) learning [23]:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t \delta_t \qquad (2)$$

where $\delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$ is the TD-error and $0 \leq \alpha_t \leq 1$ is the learning rate. It has been proven that in tabular representation, the value estimates, updated by using (2), will eventually converge to the actual state values for a fixed policy [23]. When the state space is continuous, function approximation is used, such as a neural network in our work, to learn a parameterized estimate of the value function. The update is then performed on the neural network parameters in the direction of $r_t + \gamma V(s_{t+1}; \theta^V_{t-1})$:

$$\theta^V_{t+1} = \theta^V_t + \alpha\, \delta_t \nabla_\theta V_t(s_t; \theta^V_t);\ \delta_t = y - V_t(s_t; \theta^V_t) \qquad (3)$$

where θ^V_t is the parameter vector of the neural network approximating the value function at time t and y is the sample value $r_t + \gamma V(s_{t+1}; \theta^V_{t-1})$. For an extensive review of similar methods, the reader is referred to [24].

In continuous action space, the problem is harder because there is no obvious way to decide in a given state which action leads to a state with the highest value. Van Hasselt and Wiering (2007) tackled this problem and proposed the Continuous-Actor-Critic-Learning-Automaton (CACLA) algorithm that handles both continuous state and action spaces [22]. A standard Actor-Critic RL model has the structure shown in Fig. 1, where the actor suggests an action a_t in state s_t and the critic evaluates the action outcome using the observed reward r_t and next state s_{t+1}, and based on this evaluation, the actor improves its future suggestions.

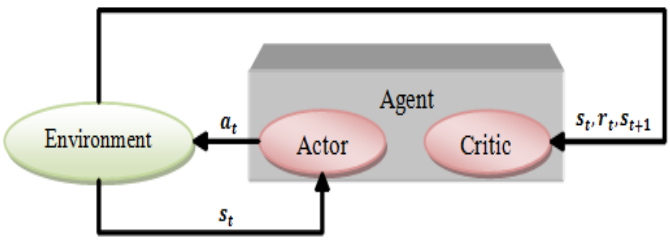

Fig. 1. Actor-Critic model. The critic and actor are responsible for action evaluation and action improvement respectively.

The basic idea behind CACLA is that if an explored action a_t has resulted in a positive change of a state value, then this

action is believed to lead to a potentially higher accumulated reward and will, therefore, be reinforced:

$$If\ \delta_t > 0 :\ increase_t\big(\pi_t(s_t, a_t)\big) \tag{4}$$

This can be implemented in continuous spaces by using two function approximators; one for the critic updated using Eq. (3) and one for the actor updated as follows:

$$If\ \delta_t > 0 :$$
$$\theta_{t+1}^{Ac} = \theta_t^{Ac} + \alpha\left(a_t - Ac_t(s_t; \theta_t^{Ac})\right)\nabla_\theta Ac_t(s_t; \theta_t^{Ac}) \tag{5}$$

where θ_t^{Ac} is the parameter vector for the actor's function approximator (a neural network here) and $Ac_t(s_t; \theta_t^{Ac})$ is the actor's output at time t given θ_t^{Ac}, which is far from a_t. No update is performed when the value is not improving because otherwise that would update towards an action that might not be better than the current estimate of the optimal action.

If an action is found to have considerably improved the value of a state, then the corresponding actor update is further magnified. This is done by first keeping a running average of the TD-error's variance:

$$var_{t+1} = (1 - \beta)\, var_t + \beta {\delta_t}^2 \tag{6}$$

and then the number of updates towards that action is determined by $\lceil \delta_t / \sqrt{var_t} \rceil$, which is relative to the number of standard deviations the target of the critic is above its output, rounded up to the next nearest integer number. This form of the algorithm is called CACLA+Var to distinguish it from the case where only a single update to the actor is performed.

For exploration in CACLA, two methods were considered by the authors: ε-greedy (selecting a uniformly-distributed random action with probability ε and the currently best-known action with probability 1- ε) and Gaussian exploration where the selected action is sampled from a Gaussian distribution with a mean at the actor's output:

$$\pi_t(s_t, a_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(a_t - Ac_t(s_t)\right)^2 / 2\sigma^2} \tag{7}$$

In the next section, we will discuss our proposed approach and show how we use CACLA as the RL algorithm within our approach.

III. ALGORITHM

The algorithm presented here is called ICAC (Intrinsically-motivated Continuous Actor-Critic). In ICAC, the agent's learning system consists of two main parts: 1) A network of predictive models of sensorimotor activity; and 2) a module of intrinsically motivated control. The full system architecture is shown in Fig. 2.

A. Network of predictive models

Much like a human infant, we want our learning agent to be able to reorganize its interaction with its environment, moving from regions where it has learned about the outcome of its motor actions to other regions where it expects to learn new

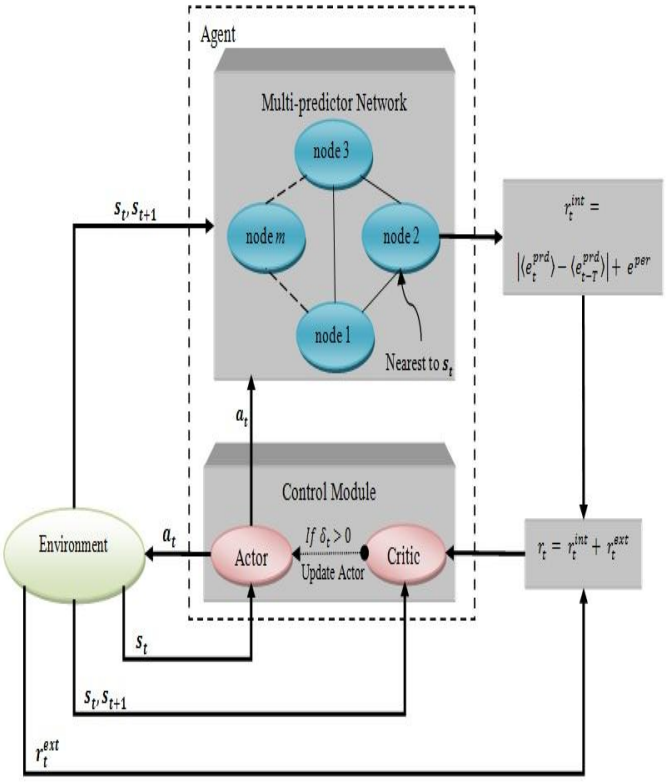

Fig. 2. ICAC architecture. The multi-predictor network adaptively forms a topology-representing network of the agent's sensory input, where each node is associated with a local predictor of the system dynamics in the region covered by the node. The agent's control module comprises an actor-critic model. The predictor of the network's nearest node to the observed state s_t predicts the next state $\hat{s}_{t+1}$ using both s_t and the taken action a_t and updates itself using the actual next state s_{t+1}. An internal reward is then generated from the perception error and the learning progress observed, combined with the external reward and sent to the critic to update its estimate of the utility of a_t accordingly.

patterns of motor activity. This can be realized by learning a number of local predictive models, which we call here activity models. To facilitate this, we equip the agent with a cognitive map-like representation that stores and integrates information about spatial connectivity among environmental regions with information about activity models.

While interacting with its environment, the agent incrementally partitions the sensory input space into regions of activity using the Instantaneous Topological Map (ITM) [25]. The ITM is an unsupervised learning method for adaptively building a topology-preserving map of the input space, specially developed for strongly correlated input, which is the case in most robotic applications where stimuli are generated by exploration along continuous trajectories. Unlike other common topology-representing networks such as the SOM [26] and the Growing Neural Gas [27], ITM is considered more computationally efficient, with the number of nodes scaling linearly with the volume of the state space, of which it provides a Delaunay triangulation, and has been successfully used in several RL problems [28],[16].

The ITM (see Fig. 3) is defined by a set of neurons/nodes i, each having a weight vector w_i and a set of neighboring nodes $N(i)$ with which it is connected by edges. A predictive model of system dynamics is assigned to each node, trained to predict the next state from the current state and agent's action.

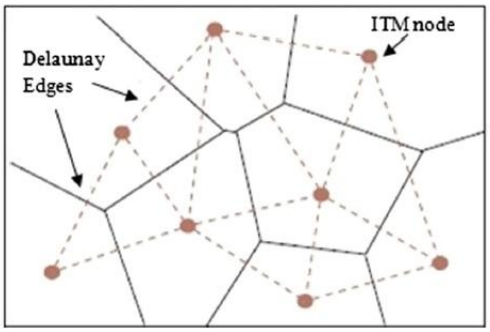

Fig. 3. ITM network. The nodes of the incrementally built ITM network are the centers of the Voronoi cells. The Delaunay triangulation (dotted lines) connects the centers of the neighboring cells. Each triangle is associated with a circle inside which no triangulation vertices can exist. Otherwise, a non-Delaunay edge is found.

Given a sample experience (s_t, a_t, s_{t+1}, r_t), the agent updates its ITM network, which starts with two connected nodes, as outlined in Algorithm 1.

Algorithm 1 Multi-predictor Network Adaptation

1: $n \leftarrow \arg\min_i \|s_t - w_i\|, n' \leftarrow \arg\min_{j, j \neq n} \|s_t - w_j\|$
2: $\Delta w_n \leftarrow \epsilon (s_t - w_n)$
3: $N(n) \leftarrow N(n) \cup \{n'\}, N(n') \leftarrow N(n') \cup \{n\}$
4: ***for*** m *in* $N(n)$ ***do***
5: ***if*** $(w_n - w_{n'}).(w_m - w_{n'}) < 0$ ***then***
6: $N(n) \leftarrow N(n) - \{m\}, N(m) \leftarrow N(m) - \{n\}$
7: ***if*** $N(m) = \{\emptyset\}$ ***then***
8: $ITM \leftarrow ITM - \{m\}$
9: ***end if***
10: ***end if***
11: ***end for***
12: ***if*** $\|s_t - w_n\| > e_{max}$ & $(w_n - s_t).(w_{n'} - s_t) > 0$ ***then***
13: $ITM \leftarrow ITM \cup \{v\}$
14: $w_v \leftarrow s_t, N(v) \leftarrow \{n\}, N(n) \leftarrow N(n) \cup \{v\}$
15: Initialize and update activity model of v
16: ***else***
17: Update activity model of n
18: ***end if***
19: ***if*** $\|w_n - w_{n'}\| < 0.5 e_{max}$ ***then***
20: $ITM \leftarrow ITM - \{n'\}$
21: ***end if***

The nearest n and the second-nearest node n' to the observed state s_t are determined based on the Euclidean distance between s_t and the weight vector of each ITM node (line 1), and n is then moved by a small rate ϵ towards s_t (line 2). An edge between n and n' is created (line 3) if no such edge exists. For all neighbors of n, we check if any edge has become invalid/non-Delaunay as a result of the recent edge creation, and if so, the invalid edge is removed and if it is the only edge of the neighbor, the neighbor is removed as well from the network (lines 4-8). If the Euclidean distance between s_t and the weight vector of n denoted by w_n is greater than a threshold e_{max} and s_t and $w_{n'}$ are on opposite sides of w_n, then a new node v is added to the ITM with a weight vector equal to s_t and an edge with n (lines 12-14). In case a new node for s_t is added, we initialize a new prediction model assigned to the new node and train it using the triple (s_t, a_t, s_{t+1}) (line 15). Otherwise, the model associated with n is updated (line 17). Finally, if the distance between w_n and $w_{n'}$ has become less than $0.5 e_{max}$, we remove s from the network (lines 19-20). The threshold e_{max} controls the growth of the ITM network and is referred to in [25] as the desired mapping resolution.

The prediction models associated with the ITM network nodes are two-layer neural networks trained online from experience data and represent a collection of local sensori-motor behaviors of the agent.

The method by which experience data is collected and the role of activity models in aiding the agent's curiosity-driven motor control will be explained in the following subsection.

B. Intrinsically motivated control

Using the currently observed state s_t of its environment, the agent is able to determine the activity region whose corresponding ITM node has the closest weight vector to s_t. The activity model associated with this node (or the newly added node in case no existing node is close enough to s_t as in line 12 of Algorithm 1) is then queried to predict the next state using s_t and the performed action a_t as an input. The difference between the true and the predicted next state, s_{t+1} and $\hat{s}_{t+1}$ respectively indicates a prediction error:

$$e_{t+1}^{prd} = \| \hat{s}_{t+1} - s_{t+1} \| \tag{8}$$

We keep an updated average of prediction error for each activity model computed over the μ recent occurrences the model was asked for predicting an action outcome:

$$\langle e_{t+1}^{prd} \rangle = \frac{\sum_{i=0}^{\mu-1} e_{t-i+1}^{prd}}{\mu} \tag{9}$$

The change in value between two temporally consecutive averages of the prediction error of a particular region carries information about the learning progress the agent is expected to make or has made to increase its ability to predict action outcomes. In other words, if the average error has increased, then there is a high potential for learning progress to be made by exploring that region. Similarly, if the average error has decreased, this means the agent has improved its prediction capabilities and experienced a learning progress. We combine this information with the perception error e^{per}, which is simply the distance between s_t and the weight vector of the nearest node of the multi-predictor network $e^{per} = \|s_t - w_n\|$, to generate an internal reward for the agent:

$$r_t^{int} = \left| \langle e_t^{prd} \rangle - \langle e_{t-T}^{prd} \rangle \right| + e^{per} \tag{10}$$

where T ($T \leq \mu$) is the time frame between two recordings of the average prediction error measured by the number of times the corresponding activity model has been queried. e^{per} is the perception error used as an incentive for visiting perceptually novel states. e^{per} is effective while the observable part of the sensory space is not fully mapped by the ITM, after which it can be neglected.

Algorithm 2 ICAC algorithm

1: Randomly initialize critic and actor networks $V(s;\theta^V)$ and $Ac(s;\theta^{Ac})$ with weights θ^V and θ^{Ac}
2: Initialize the variance of the TD-error: $Var_{t=0} \leftarrow 1$
3: ***for*** $episode = 1$ *to* M ***do***
4: Receive initial state s_0
5: ***for*** $step = 1$ *to* N ***do***
6: Select action a_t from a Gaussian distribution centered at the actor's output $Ac(s_t;\theta_t^{Ac})$
7: Execute a_t and observe reward r_t^{ext} and next state s_{t+1}
8: Update the multi-predictor network using (s_t, a_t, s_{t+1}), as detailed in Algorithm 1
9: Compute internal reward r_t^{int}, as in Eq. (10)
10: $r_t \leftarrow r_t^{int} + r_t^{ext}$
11: Set critic target: $y \leftarrow r_t + \gamma V(s_{t+1};\theta_{t-1}^V)$
12: Update critic to minimize the cost (see Eq. (3)): $J_c = \frac{1}{2}\left(y - V(s_t;\theta_t^V)\right)^2$
13: $\delta_t \leftarrow y - V(s_t;\theta_t^V)$
14: $Var_{t+1} \leftarrow (1-\beta)Var_t + \beta{\delta_t}^2$
15: ***if*** $\delta_t > 0$ ***then***
16: Update actor $\left\lfloor \frac{\delta_t}{\sqrt{var_t}} \right\rfloor$ times to minimize the cost (see Eq. (5)): $J_a = \frac{1}{2}\left(a_t - Ac(s_t;\theta_t^{Ac})\right)^2$
17: ***end if***
18: ***end for***
19: ***end for***

This self-generated reward acts as a curiosity signal for the agent to try actions that maximize its learning progress and thus enabling it to learn the desired task more efficiently. The reason is that, as the agent's ability to model transition dynamics in a region of its sensory space improves, it has come to understand that region better, which is important to perform tasks in the environment. In order to introduce this reward to the agent's control system, we use it in combination with an external reward from the environment (if any):

$$r_t = r_t^{int} + r_t^{ext} \tag{11}$$

The combined reward r_t is then passed to the critic of the CACLA control module to update its estimate of the value of s_t and update the actor when the resulting TD-error is positive, as shown in Fig. 2.

The pseudo code of the proposed ICAC algorithm is presented in Algorithm 2. Main changes to the standard CACLA+Var algorithm are shown in lines 8-10, including the adaptation of the multi-predictor network and its predictive models. We approximate the critic and actor by a neural network with two layers trained to minimize the relative costs J_c and J_a respectively.

IV. EVALUATION

In the following, we describe the experimental setup and present two robotic experiments for learning control tasks with increasing difficulty. All parameter values were empirically determined after preliminary testing on the environments considered. The experiments were run using a discount factor of 0.9. This value did not correlate with the performance. All actions were drawn from a Gaussian distribution with a mean at the actor's output and standard deviation of 0.1. Both the actor and critic were represented by two-layer feedforward neural networks with 12 hidden neurons. Different numbers made no significant difference to the results. We used hyperbolic tangent and linear activation functions for the hidden and output layers respectively. The learning rate α used to update the actor and critic networks' weights was set to 0.01. For computing the TD-error's variance, we used a factor β of 0.01. Varying α and β did not affect the performance adversely.

The node creation threshold e_{max} of the ITM-based multi-predictor network was set to 0.9. Smaller values were found to increase the computation time without considerable performance gain. The learning step ϵ was set to 10^{-4}. All predictive models of the multi-predictor network were two-layer feedforward neural networks with the same number of hidden neurons and learning rate as the actor and critic networks and with hyperbolic tangent activation in the hidden and output layers. The hyperbolic tangent at the output ensures that the input and output states are in $[-1, 1]$ and that the prediction error remains bounded. The number μ of successive predictions of a predictive model, used to average the corresponding prediction errors, was 40. The time frame T used in deriving the internal reward was 20. All inputs to the neural networks were normalized to the interval $[-1, 1]$. Actor outputs were bounded to $[-10, 10]$ in degrees to only allow learning of action sequences as opposed to single-step actions to the goal.

A. Reaching with 2-DoF robotic arm

In this experiment, we test our ICAC algorithm on a simple control task of reaching a variable goal in 2D space with a 2-DoF robotic arm, as shown in Fig. 4. The state representation used in the actor and critic networks is a four-dimensional real-valued vector with two components corresponding to the current joint values of the arm in degrees and another two corresponding to the Cartesian coordinates of the current goal position. The actions are two-dimensional real-valued vectors, containing the angular changes of the joints. The reward from the environment after taking an action is defined as follows:

$$r_t^{ext} = \begin{cases} +50 & when\ reaching\ the\ current\ goal\ zone, \\ 0 & otherwise. \end{cases}$$

Each of the arm's two links is one unit length and the goal zone radius is 0.3 unit length.

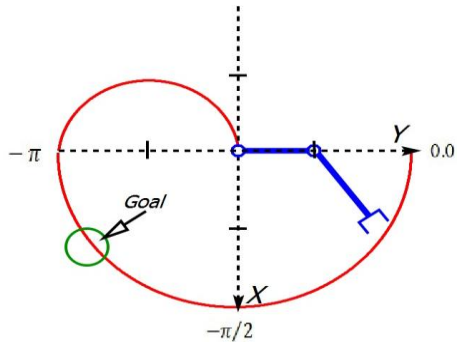

Fig. 4. Goal reaching with a 2-DoF robotic arm. The red curve specifies the reachable workspace for the arm. The green circle is the current goal zone.

Learning is performed over 1000 episodes, and the agent is given a maximum of 50 timesteps to reach the goal, after which the agent resets to a random initial configuration, and a new random goal is generated. The results are averaged over 20 simulations. We compare our proposed algorithm ICAC to CACLA and CACLA+Var. Fig. 5 shows the average external reward obtained per episode of the three algorithms.

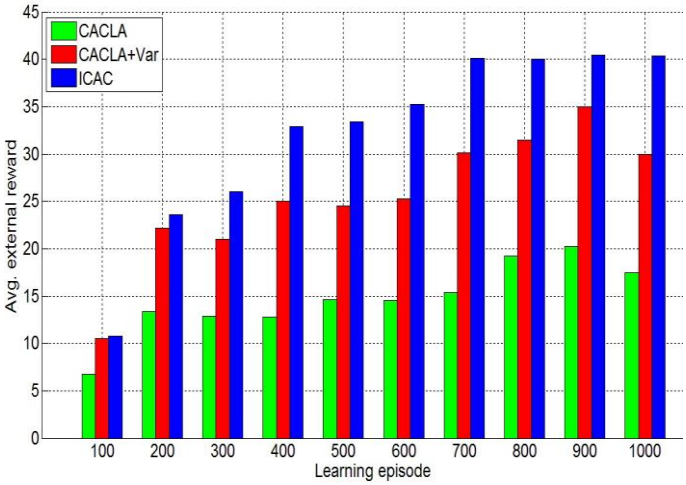

Fig. 5. Goal reaching with a 2-DoF robotic arm. Average external reward.

Although an external reward was provided to the agent during learning only when reaching the current goal zone, the ICAC agent was able to reach each new goal position more often than the agent running any of the other two algorithms. As expected, CACLA+Var showed slightly higher performance than CACLA. Similarly, Fig. 6 compares the three algorithms in terms of the average number of steps taken to reach the randomly generated goal. While CACLA and CACLA+Var converged to a policy of about 35 and 20 actions toward the goal respectively, ICAC continued to learn and converged to a better action policy of slightly less than 10 actions on average toward the goal.

The reason CACLA does not quickly find optimal policy, as seen in Fig. 5 and Fig. 6, is that once a goal is reached, all future actor outputs will be largely influenced by the first action sequence found to lead to the previously reached goal and will hardly suggest other action sequences to be taken to reach other new goal positions. Conversely, the actor of ICAC mostly chooses actions that maximize the learning progress. These actions keep changing as the perception error and the predictive models evolve, allowing the discovery of new policies with higher rewards and fewer actions.

B. Reaching with 4-DoF NICO arm

We evaluate here the three algorithms on our humanoid robot NICO [29] in 3D space. The task is to learn to move a 4-DoF robotic arm to reach the desired goal region. The experiment was run in the V-REP robot simulation environment, as shown in Fig. 7.

The joints considered in the experiment are shown in Table I. The states and actions are four-dimensional real-valued vectors of joint angles and angular changes respectively. The input to the actor and critic networks are 7-dimensional real-valued vectors consisting of the 4D state representation and the Cartesian coordinates of the current goal position.

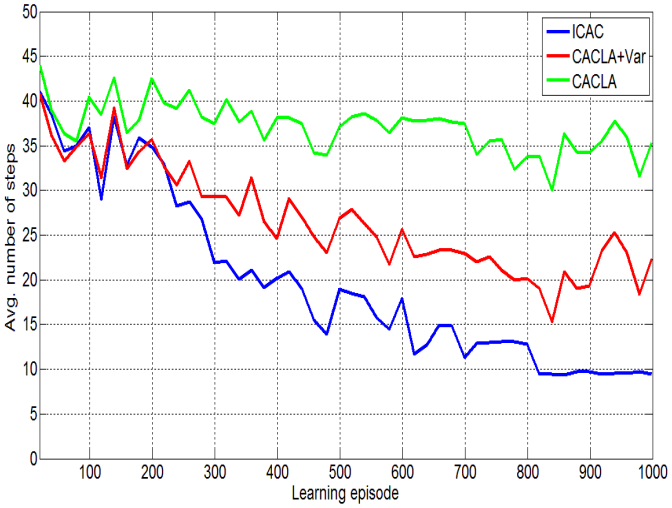

Fig. 6. Goal reaching with a 2-DoF robotic arm. Average number of steps to the goal. The average over 20 episodes is shown for readability.

We use a dummy point in the right lower arm to serve as the end-effector. For the goal region, a radius of 12.5% of the robot's arm length is used. Again, we do not provide any external rewards until the robot reaches the goal region in which it receives a positive external reward of 100:

$$r_t^{ext} = \begin{cases} +100 & when\ reaching\ the\ goal\ zone, \\ 0 & otherwise. \end{cases}$$

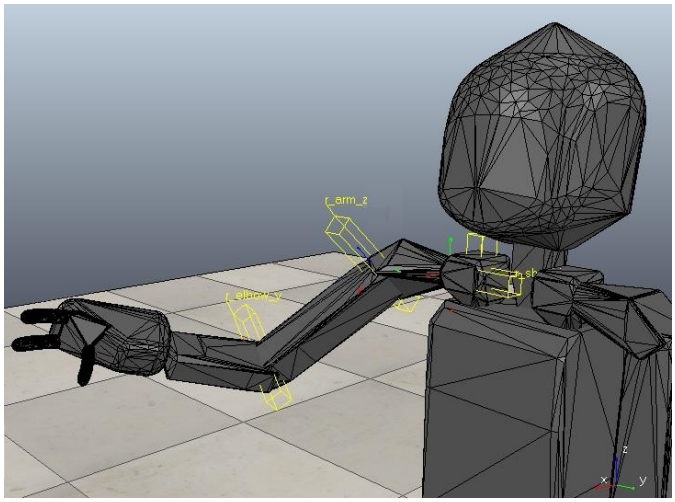

Fig. 7. Simulated NICO robot. Four joints in the right arm are used to learn to perform a reaching task in the 3D workspace.

Each simulation experiment consists of 15000 learning episodes in which the robot is given 50 action attempts to reach the goal before it is set to a random rest configuration, then a new random goal position is generated.

We averaged the results over 20 simulations. The average external reward obtained by the robot running each of the three algorithms is shown in Fig. 8.

The CACLA algorithms were able to reach only a maximum average reward of around 60 after 4500 episodes, indicating that they failed to reach the goal in eight out of 20 simulations (60% successes). In contrast, the rate of successful simulations of the ICAC algorithm continued to increase with the learning episodes, reaching over 90% by the end of the learning process.

Table I. List of the joints considered in the second experiment.

Joint	Description	Angle limit (in degrees)
r_shoulder_z	rotates around the z-axis of the local frame attached to the right shoulder.	$[-100, 125]$
r_shoulder_y	rotates around the y-axis of the local frame attached to the right shoulder.	$[-180, 179]$
r_arm_z	rotates around the z-axis of the local frame attached to the right arm.	$[-140, 75]$
r_elbow_y	rotates around the y-axis of the local frame attached to the right elbow.	$[-100, 100]$

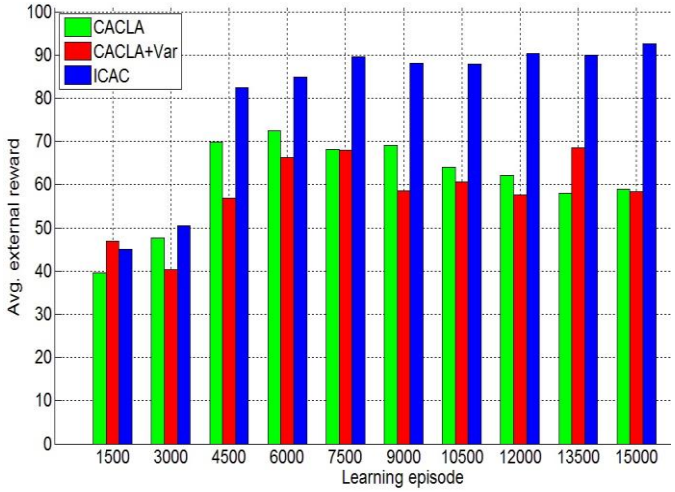

Fig. 8. Goal reaching with a 4-DoF robotic arm. Average external reward.

Regarding the average number of steps to the goal, CACLA and CACLA+Var learned a policy of 30 and 20 steps respectively, whereas the ICAC needed only around five steps on average to reach the goal with the number sharply decreasing over the first half of the learning episodes until convergence, as illustrated in Fig. 9.

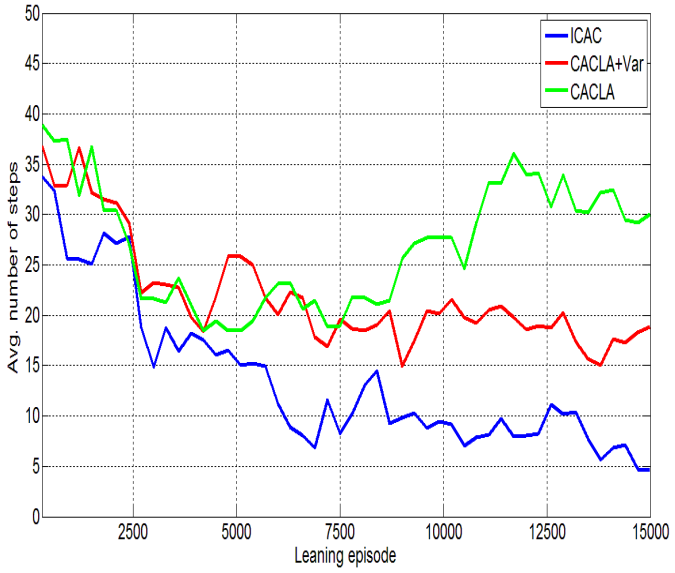

Fig. 9. Goal reaching with a 4-DoF robotic arm. Average number of steps to the goal. The average over 300 episodes is shown for readability.

V. CONCLUSION

In this paper, we present ICAC as an algorithm for reinforcement learning of continuous control. This algorithm integrates unsupervised learning of sensory representations with online learning of predictive models of motor dynamics to generate an intrinsic reward for guiding the exploration of an autonomous agent. The intrinsic rewards reflect the learning progress of the agent and are combined with external rewards from the environment to learn control tasks more effectively and rapidly than when only external rewards are used, which is vital for open-ended developmental learning systems. The experimental results show that ICAC achieves better performance when compared to other related continuous actor-critic algorithms.

One of the limitations of our approach is that although it ultimately learns better control policies, it requires intensive exploration in the early learning episodes which is necessary for learning accurate and useful predictive models. A potential alternative would be to provide the agent with good initial policies by imitation learning methods to reduce the action search space. Another issue is the dimensionality of the sensory space if a real sensor like a camera is used. This can be mitigated by passing low-dimensional feature embedding from a deep autoencoder to the ITM map rather than directly using the raw sensory input.

ACKNOWLEDGMENT

We gratefully acknowledge the support by the DAAD German Academic Exchange Service (Funding Programme No. 57214224).

REFERENCES

[1] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research,* vol. 4, pp. 237-285, 1996.

[2] 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 and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[3] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proceedings of the seventh International Conference on Machine Learning (ICML)*, 1990.

[4] R. I. Brafman and M. Tennenholtz., "R-MAX a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research,* vol. 3, no. Oct, pp. 213-231, 2002.

[5] L. Lihong, M. L. Littman and C. R. Mansley, "Online exploration in least-squares policy iteration," in *The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Budapest, Hungary, 2009.

[6] I. Osband, C. Blundell, A. Pritzel and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances In Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, 2016.

[7] S. Singh, A. G. Barto and N. Chentanez, "Intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, British Columbia, Canada, 2005.

[8] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Transactions on Autonomous Mental Development,* vol. 2, no. 3, pp. 230-247, 2010.

[9] P. Y. Oudeyer, F. Kaplan and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on*

Evolutionary Computation, vol. 11, no. 2, pp. 265-286, 2007.

[10] M. Luciw, V. Graziano, M. Ring and J. Schmidhuber, "Artificial curiosity with planning for autonomous perceptual and cognitive development," in *Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, Frankfurt, 2011.

[11] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck and P. Abbeel, "VIME: variational information maximizing exploration," in *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, 2016.

[12] S. Mohamed and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems (NIPS)*, Montréal, Canada, 2015.

[13] B. C. Stadie, S. Levine and P. Abbeel, "Incentivizing exploration in reinforcement rearning with deep predictive models," arXiv preprint arXiv:1507.00814, 2015.

[14] T. Hester and P. Stone, "Intrinsically motivated model learning for developing curious robots," *Artificial Intelligence,* 2015.

[15] Ş. Özgür and A. G. Barto, "An intrinsic reward mechanism for efficient exploration," in *International Conference on Machine learning (ICML)*, Pittsburgh, Pennsylvania, USA, 2006.

[16] M. B. Hafez and C. K. Loo, "Topological Q-learning with internally guided exploration for mobile robot navigation," *Neural Computing and Applications,* vol. 26, no. 8, pp. 1939-1954, 2015.

[17] K. E. Twomey and G. Westermann, "A neural network model of curiosity-driven infant categorization," in *Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, Rhode Island, USA, 2015.

[18] C. Kidd, S. T. Piantadosi and R. N. Aslin, "The Goldilocks effect in infant auditory attention," *Child Development,* vol. 85, no. 5, pp. 1795-1804, 2014.

[19] M. Rolf, J. J. Steil and M. Gienger, "Goal babbling permits direct learning of inverse kinematics," *IEEE Transactions on Autonomous Mental Development,* vol. 2, no. 3, pp. 216-229, 2010.

[20] A. Baranes and P. Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems,* vol. 61, no. 1, pp. 49-73, 2013.

[21] 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," in *International Conference on Learning Representations (ICLR)*, 2016.

[22] H. Van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007.

[23] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning,* vol. 3, no. 1, pp. 9-44, 1988.

[24] D. Bertsekas and J. N. Tsitsiklis, Neuro-dynamic programming, Belmont, MA: Athena Scientific, 1996.

[25] J. Jockusch and H. Ritter, "An instantaneous topological mapping model for correlated stimuli," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN 1999)*, Washington, DC, 1999.

[26] T. Kohonen, Self-organization and associative memory, New York: Springer Berlin Heidelberg, 1989.

[27] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems,* vol. 7, p. 625–632, 1995.

[28] A. P. S. Braga and A. F. Araujo, "Influence zones: A strategy to enhance reinforcement learning," *Neurocomputing,* vol. 70, no. 1-3, pp. 21-34, 2006.

[29] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich and S. Wermter, "NICO -- Neuro-Inspired COmpanion: A Developmental Humanoid Robot Platform for Multimodal Interaction," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017 accepted.