Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Intrinsically Motivated Actor-Critic for Robot Motor Learning

Dissertation

Dissertation submitted to the University of Hamburg
with the aim of achieving a doctoral degree at the
Faculty of Mathematics, Informatics and Natural Sciences,
Department of Informatics.

Muhammad Burhan Hafez

Hamburg, 2020

Submitted:
February 13th, 2020

Day of oral defense:
May 7th, 2020

The following evaluators recommend the admission of the dissertation:

Prof. Dr. Stefan Wermter (advisor)
Department of Informatics, Universität Hamburg, Germany

Prof. Dr. Victor Uc-Cetina (reviewer)
Department of Informatics, Universität Hamburg, Germany

Prof. Dr. Jianwei Zhang (chair)
Department of Informatics, Universität Hamburg, Germany

To my parents, Ayman and Najah.

Abstract

Learning sensorimotor skills from trial and error in unknown environments is an important ability for autonomous agents. Reinforcement learning (RL) is a powerful approach to provide this ability without manually programming the desired behavior or requiring any prior knowledge by learning a control policy–a direct mapping from a raw sensory input to a raw motor output that optimizes the task performance. In recent years, deep RL has been used to learn this mapping from self-collected experience data by utilizing deep neural networks as function approximators. However, the performance of deep RL critically depends on the chosen exploration strategy. Random, undirected exploration is impractical in real-world robot learning where spending additional training time exploring known parts of the robot's environment cannot be afforded and converging to an optimal policy with a minimal number of environmental interactions is necessary. Deep RL also suffers from poor sample efficiency as it requires large amounts of self-collected training data to adjust the large number of learning parameters the deep networks typically have. This fundamentally limits how quickly a robot can learn useful control policies. Another issue inevitable in complex domains is using imperfect predictive models of the environment for planning actions or simulating experiences, which results in a compounding of prediction errors when making multiple-step predictions with a learned model, leading to poor task performance.

In this thesis, we propose behaviorally and neurally plausible approaches to better understand, analyze, and address these different challenges associated with improving deep RL for robot sensorimotor learning. We first present a directed, data-efficient exploration strategy, inspired by sensorimotor development in infants, which provides an intrinsic reward based on the progress in learning local world models, informatively shifting the interactions from well-explored to less-explored regions of the world. We then introduce a novel deep architecture for learning low-dimensional state representations that optimize a joint supervised reward prediction and unsupervised input reconstruction loss. The learned representation is used as input to the policy and the world models whose predictions are used in deriving the intrinsic reward, effectively improving the sample efficiency of learning real-world pixel-level policies. We next show how the learning progress-based intrinsic reward can be used as an estimate of the reliability of model predictions to determine when to perform planning under the model and adaptively arbitrate between model-based and model-free control in dual-system RL. We also leverage the learned model by augmenting the training set of real experiences with imagined experiences generated with the model over regions where the model is reliable and use the intrinsic motivation to collect experience data that improves the model. Finally, we propose a dual-system motor learning approach that integrates arbitration with imagination and enables an adaptive-length model rollout for plan optimization during model-based control, and evaluate it on learning vision-based robotic grasping in simulation and the real world. The experimental results show that our approach learns better vision-based grasping policies than baseline and state-of-the-art methods in dense- and sparse-reward environments.

Zusammenfassung

Das Erlernen sensomotorischer Fähigkeiten durch Versuch und Irrtum in unbekannten Umgebungen ist eine wichtige Fähigkeit für autonome Agenten. Reinforcement Learning (RL) ist ein leistungsfähiger Ansatz, um diese Fähigkeiten zu erlangen ohne das gewünschte Verhalten manuell zu programmieren oder Vorwissen zu nutzen. Beim Reinforcement Learning wird eine Control Policy gelernt, eine direkte Abbildung von einem unvorverarbeitetem sensorischen Input zu einem motorischen Output, die eine gegebene motorische Aufgabe optimal ausführt. In den letzten Jahren wurde tiefes RL verwendet, um diese Abbildung auf Grundlage von selbst-gesammelten Erfahrungsdaten mit tiefen neuronalen Netzen als Funktionsapproximatoren zu lernen. Jedoch hängt die Performanz von tiefem RL entscheidend von der gewählten Explorationsstrategie ab. Zufällige, ungerichtete Exploration ist beim Lernen mit Robotern in der realen Welt unpraktisch. Unnötige Trainingszeit, in der bekannte Teile der Roboterumgebung erforscht werden, sollte vermieden werden. Eine optimale Policy sollte mit einer minimalen Anzahl von Umweltinteraktionen erlernt werden. Tiefes RL leidet zudem unter einer schlechten Dateneffizienz; große Mengen an selbst gesammelten Trainingsdaten sind notwendig um die typischerweise große Anzahl von Parametern in den tiefen Netzwerken zu lernen. Diese schränkt grundsätzlich ein, wie schnell ein Roboter nützliche Policies erlernen kann. Ein weiteres Thema, dass in komplexen Umgebungen unvermeidlich ist, sind unvollkommene Vorhersagemodelle für die Planung von Aktionen oder die Simulation von Erfahrungen. Vorhersagefehler können sich bei Mehrschritt-Vorhersagen mit gelernten Modellen aufsummieren und zu einer schlechten Performanz führen.

In dieser Arbeit präsentieren wir psychologisch und neurologisch plausible Ansätze zum besseren Verständnis, zur Analyse und zur Lösung dieser verschiedenen Herausforderungen des tiefen sensomotorischen RL für Roboter. Wir stellen zunächst eine gezielte, dateneffiziente Explorationsstrategie vor, inspiriert von der sensomotorischen Entwicklung von Säuglingen, die eine intrinsische Belohnung auf Grundlage der Fortschritte beim Erlernen lokaler Weltmodelle nutzt. Diese Strategie lenkt die Exploration von gut erforschten auf weniger bekannte Bereiche der Welt. Dann stellen wir eine neue tiefe Architektur für das Lernen von niedrig-dimensionalen Zustandsrepräsentationen vor, welche gleichzeitig eine überwachte Belohnungsvorhersage und eine unüberwachte Rekonstruktion des Inputs realisiert. Die erlernte Repräsentation wird als Input für die Policy und die Weltmodelle genutzt, deren Vorhersagen in die Bestimmung der intrinsischen Belohnung einfließen und somit die Dateneffizienz beim Erlernen von Policies auf Pixelebene verbessern. Als Nächstes zeigen wir, wie die auf dem Lernfortschritt basierende intrinsische Belohnung als Schätzung der Zuverlässigkeit der Modellvorhersagen genutzt werden kann, um zu bestimmen, wann Planung mit diesem Modell durchgeführt werden soll um somit adaptiv zwischen modellbasierter und modellfreier Kontrolle in einem dualen RL-System zu wechseln. Wir nutzen die gelernten Modelle ebenfalls, um die Trainingsdaten aus realen Erfahrungen durch imaginäre Erfahrungen in jenen Regionen zu ergänzen, in denen das Modell hinreichend zuverlässig ist. Gleichzeitig führt die intrinsische Motivation wiederum dazu, dass bevorzugt solche Erfahrungen gesammelt werden, die das Modell verbessern. Schließlich schlagen wir einen Ansatz für motorisches Lernen mit einem dualen System vor, welches einen adaptiven Wechsel zwischen modellbasierter und modellfreier Kontrolle mit imaginären Erfahrungen verbindet und modellbasierte Planung mit einer adaptiven Anzahl von Planungsschritten ermöglicht. Wir evaluieren den Ansatz mit robotischen Experimenten zu sichtbasiertem Greifen in simulierten und realen Umgebungen. Die experimentellen Ergebnisse zeigen, dass unser Ansatz eine bessere, auf sichtbasiertem Greif-Policy lernt als Basis- und modernste Methoden in Umgebungen mit häufigen als auch spärlichen Belohnungen erlernen kann.

Contents

1 Introduction 1

1.1 Problem Statement and Research Objectives 1
1.2 Actor-Critic and Intrinsic Motivation 2
1.2.1 Actor-Critic Reinforcement Learning 3
1.2.2 Intrinsic Motivation for Reinforcement Learning 6
1.3 Summary of Contributions 7
1.4 Thesis Organization 10

2 Spatially and Temporally Local Learning Progress 13

2.1 Introduction 13
2.2 Growing Self-Organization of Sensory Space 16
2.3 Intrinsically Motivated Continuous Actor-Critic 17
2.4 Experiments 19
2.4.1 Reaching with 2-DoF Robotic Arm 21
2.4.2 Reaching with 4-DoF NICO Arm 23
2.5 Conclusion 25

3 Intrinsically Motivated Actor-Critic with Efficient State Representations 27

3.1 Introduction 27
3.2 ICAC with Slow Features 29
3.2.1 Hierarchical Slow Feature Analysis 30
3.2.2 SFA-based ICAC 32
3.2.3 Experiments 34
3.3 ICAC with Deep Convolutional Autoencoder 36
3.3.1 Jointly Optimized Representations with Actor-Critic-Autoencoder Architecture 36
3.3.2 Deep ICAC 38
3.3.3 Experiments 40
3.4 Conclusion 54

4 Adaptive Arbitration between Model-based and Model-free Control 57

4.1 Introduction 57
4.2 Dual-System Motor Learning 58
4.3 Dual-System Deep RL for Continuous Control 60
4.4 Curious Meta-Controller 63
4.4.1 Model-free Control with Off-policy Actor-Critic 63
4.4.2 Model-based Control with MPC 65
4.4.3 Meta-Decision Making 66
4.5 Experiments 68

4.5.1 Vision-based Robotic Reaching 70
4.5.2 Vision-based Robotic Grasping 71
4.6 Conclusion 74

5 Learning-Adaptive Imagination in Latent Space 77
5.1 Introduction 77
5.2 Experience Imagination in Deep RL 79
5.3 Learning-Adaptive Imagination 80
5.3.1 Latent Representation Learning 81
5.3.2 Latent-Space Self-Organization 81
5.3.3 Latent-Space Experience Imagination 83
5.4 Experiments 86
5.5 Conclusion 90

6 Integrated Imagination-Arbitration in Self-Organized Latent Space 93
6.1 Introduction 93
6.2 Intrinsically Motivated Meta-Controller 94
6.2.1 Local Reliability Estimation 95
6.2.2 Reliability-based Arbitration 96
6.3 Integrated Imagination-Arbitration Learning Framework 98
6.4 Experiments 100
6.4.1 Evaluation in Simulation 100
6.4.2 Evaluation on a Real Robot 106
6.5 Conclusion 108

7 Conclusion 109

A List of Abbreviations 113

B Supplementary Algorithms 115

C Publications Originating from this Thesis 119

D Acknowledgements 121

Bibliography 123

List of Figures

2.1 ITM network 16
2.2 ICAC learning system 18
2.3 Goal reaching with 2-DoF robotic arm 21
2.4 Average extrinsic reward on random goal reaching with 2-DoF arm 22
2.5 Average number of steps to the goal 22
2.6 Simulated NICO robot used in the reaching task 23
2.7 Average extrinsic reward on random goal reaching with 4-DoF arm 24
2.8 Average number of steps to the goal on random goal reaching with 4-DoF 25

3.1 Hierarchical SFA network 32
3.2 SFA-based ICAC system 33
3.3 The 3D simulation environment with a 3-DoF robotic arm 34
3.4 Learning curves of ICAC and the baselines 35
3.5 The proposed learning architecture 37
3.6 Deep ICAC learning system 40
3.7 The V-REP simulation environment for the task of learning-to-reach 44
3.8 Performance curves on the robotic learning-to-reach task 45
3.9 The V-REP simulation environment in the learning-to-grasp experiment 46
3.10 The raw motor output and raw sensory input 47
3.11 Learning curves on the robotic learning-to-grasp task 48
3.12 NICO experimental setup during learning 49
3.13 The image obtained from the top-mounted camera 50
3.14 Learning curves of Deep ICAC on the simulated and physical NICO 52

4.1 Model-free control system 64
4.2 Model-based control system 66
4.3 Curious Meta-Controller (CMC) 68
4.4 V-REP simulation environment for random target reaching 70
4.5 Learning curves of DDPG and CACLA with and without CMC on random target reaching from pixel input 71
4.6 Vison-based grasp learning experiment 72
4.7 Learning curves of DDPG and CACLA with and without CMC on robotic grasping from pixel input 73

4.8 The performance of the latent dynamics model of CMC 74
4.9 Learning curves of CACLA+CMC on the grasping task with sparse rewards for different planning horizons 75

5.1 Actor-critic and dynamics model architectures 82
5.2 Overview of our learning-adaptive imagination system 85
5.3 Motor output and sensory input for the grasp learning task 88
5.4 Learning curves on robotic grasp learning from pixels 89
5.5 Learning curves of Deep ICAC+LA-Imagination for different values of maximum imagination depth 90

6.1 Adaptive-length model rollout for model-based control 97
6.2 Intrinsically motivated meta-controller 98
6.3 Integrated Imagination-Arbitration (I2A) framework 101
6.4 V-REP-simulated grasp-learning experiment 103
6.5 Learning curves of off-policy CACLA and DDPG with and without IM2C on robot grasp learning from pixel input 104
6.6 Learning curves of DDPG+IM2C, DDPG+CMC, and MVE-DDPG on robot grasp learning from pixel input 105
6.7 Learning curves of DDPG+I2A, DDPG+IM2C, and DDPG+LA-Imagination on robot grasp learning from pixel input 106
6.8 NICO experimental setup during a grasping test trial 107

List of Tables

2.1 The joints considered in the reaching experiment with 4-DoF NICO arm 24

3.1 Learning speed and final performance on vision-based robotic reaching with 3-DoF arm 35
3.2 Comparison between the number of learning parameters of the different deep architectures used in the experiments 42
3.3 Learning statistics in the learning-to-reach experiment 45
3.4 Test results of running Deep ICAC using the networks trained on the physical NICO 52

5.1 Learning speed and convergence of the algorithms 89

6.1 Summary statistics of the simulation results for different settings 104
6.2 Success rate of the trained policy networks on the real robot 108

Chapter 1

Introduction

Reinforcement Learning (RL) enables artificial agents to learn how to perform sequential decision-making tasks from trial-and-error experience, without manually programming the desired behavior. This involves online learning of a control policy–a direct mapping from a raw and often high-dimensional sensory input to a raw motor output that optimizes the task performance. In recent years, deep RL has been used to learn this mapping from self-collected experience data by utilizing deep neural networks as function approximators and has shown a great success in learning control behavior across different domains, achieving superhuman performance in a variety of Atari and board games (Mnih et al., 2015; Silver et al., 2016) and facilitating the acquisition of complex robotic manipulation skills (Levine et al., 2015; Levine et al., 2016; Gu et al., 2017).

1.1 Problem Statement and Research Objectives

The performance of deep RL critically depends on the chosen exploration strategy, particularly in continuous control tasks. Existing approaches to exploration are mostly based on adding noise to the policy output and therefore only search locally in the action space to improve the policy. Such random, undirected exploration is impractical in real-world robot learning where spending additional training time exploring known parts of the robot's environment cannot be afforded and converging to an optimal policy with a minimal number of environmental interactions becomes necessary. Deep RL algorithms also suffer from poor sample efficiency, which is highly challenging in visuomotor control asks where real-time constraints and noisy observations are common. This fundamentally limits how quickly a robot can learn promising control policies.

While these issues commonly exist for any deep RL approach to robot motor learning, model-based deep RL has another issue inevitable in complex domains, which is the use of an imperfect predictive model of the environment for action planning and experience generation. This is because model-based RL requires a reliable model. Making multiple-step predictions with a learned predictive model therefore leads to a compounding of prediction errors, eventually resulting in poor task performance.

In this thesis, we propose behaviorally and neurally plausible approaches to better understand, analyze, and address these different challenges associated with improving deep RL for robot sensorimotor learning. Particularly, we aim to achieve the following objectives:

1. To provide a directed exploration strategy that enables RL agents to efficiently learn continuous control tasks in dense and sparse reward environments.
2. To reduce the sample complexity of learning robotic control policies directly from raw visual input in simulation and in the real world.
3. To design an unbiased estimate of the reliability of model predictions for adaptively deciding when to switch between model-based and model-free control in dual-system RL.
4. To generate imagined experiences under a learned model of the environment dynamics based on the model reliability and use them as additional training data.
5. To develop a dual-system motor learning approach that enables an adaptive-length model rollout for plan optimization during model-based control.

1.2 Actor-Critic and Intrinsic Motivation

In this section, we will provide background on model-based and model-free RL and the use of neural networks as function approximators for RL in continuous state-action spaces, followed by an overview of actor-critic methods in RL. We will then discuss the concept of intrinsic motivation in the context of RL and review different approaches to designing intrinsic reward functions.

1.2.1 Actor-Critic Reinforcement Learning

We consider a standard RL problem where an agent interacts with a fully observable environment over a series of time steps. At every time step t, the agent observes the state of the environment s_t and performs an action a_t that depends on the current state s_t. At the following time step t+1, the agent receives a scalar reward r_{t+1} from the environment and observes a new state s_{t+1}, and the process is repeated. The goal is to find an optimal action policy that maximizes the long-term reward. The RL problem is typically formalized using Markov decision processes (MDPs). A finite-horizon MDP is described as a tuple $\{S, A, d, P, r, \gamma\}$, where S is the state space, A is the action space, d is the distribution of start states, P is the state transition distribution P: $(S \times A) \times S \rightarrow [0,1]$, r is the reward function $r : S \times A \rightarrow \mathbb{R}$, and γ is a discount factor $\gamma \in [0,1]$. The discounted sum of future rewards over time horizon T specifies the return $R_t = \sum_{i=t}^{T-1} \gamma^{i-t}\, r(s_i, a_i)$. The agent's behavior is defined by an action policy π that maps from states to probability distribution over actions $\pi : S \rightarrow P(A)$. The value of a state given a policy is defined as the expected return starting from that state and following the policy:

$$V^{\pi}(s) = \mathbb{E}_{a_i \sim \pi, s_i \sim p}[R_t \mid s_t = s].$$

The solution to the MDP is the optimal policy π^* that maximizes the expected return:

$$\pi^* = arg \max_{\pi} \ \mathbb{E}_{s_0 \sim d}\, [V^{\pi}(s_0)]. \tag{1.1}$$

In discrete state and action spaces, Equation 1.1 is solved by first defining an action-value function, called Q-funcation, that computes the expected value of taking a particular action at a particular state and following a fixed policy π thereafter, instead of the expected value of being at a particular state computed by V. The optimal Q-function Q^* associated with the optimal policy π^* can be recursively defined via the Bellman optimality equation:

$$Q^*(s, a) = \mathbb{E}_{s\prime \sim p(.|s,a)} \left[r(s, a) + \gamma \max_{a\prime \in A} Q^*(s', a') \right], \tag{1.2}$$

where s' is the subsequent state after taking action a at state s. The solution to the Bellman optimality equation is unique and can be found by solving the system of equations for all state-action pairs, as given by Equation 1.2. Once the solution is obtained, determining the optimal policy is straightforward: at each state, take the action with the highest Q-value. When the true transition dynamics P and reward function r are known to the agent, dynamic programming methods are guaranteed to converge to the optimal Q-function $Q^*(.,.)$ and, therefore, the optimal policy π^* after enough iterations. For example, value iteration is a dynamic programming method that randomly initializes the Q-function and then recursively updates it until convergence:

$$Q(s,a) = \mathbb{E}_{s\prime\sim p(.|s,a)}\left[r(s,a) + \gamma \max_{a\prime\in A} Q(s',a')\right]. \tag{1.3}$$

However, if the agent has no access to true dynamics and reward functions, then one possibility is to learn from experience an approximate model of the environment that estimates the next state and reward given the current state and action, and then solve for the optimal policy using dynamic programming. This is known as model-based RL. Model-free RL, on the other hand, does not require a model of the environment but instead learns to estimate the value function directly from experience by incrementally updating its estimate of state values or action values (Q-values) after each interaction with the environment:

$$Q(s,a) = Q(s,a) \ + \ \alpha\left[r + \ \gamma \max_{a\prime\in A} Q(s',a') - Q(s,a)\right], \tag{1.4}$$

where r is the observed reward and α is the learning rate ($0 < \alpha \ \leq 1$). This type of learning is called Temporal-Difference (TD) learning, since it updates the value estimate of a state or a state-action pair based on the difference between two consecutive estimations; the expected value and the obtained value, which are $Q(s,a)$ and $r + \gamma \max_{a\prime\in A} Q(s',a')$, respectively, as shown in Equation 1.4. This difference is referred to as the TD error, $\delta = \ r + \ \gamma \max_{a\prime\in A} Q(s',a') \ - Q(s,a)$. Q-learning is a commonly used model-free RL algorithm that learns an approximation of the optimal Q-function using TD learning (see Equation 1.4). It is an off-policy algorithm, meaning that the agent can follow an arbitrary exploration policy and is guaranteed to find the optimal policy given that every state-action pair is visited infinitely often.

In continuous state spaces, it is infeasible to learn about every state and use a tabular representation of the value function to store and update state or state-action values and hence generalization of learning across states is necessary. Deep Q-learning (Mnih et al., 2013; Mnih et al., 2015) is an example of performing this generalization by using a deep neural network, called the deep Q-network (DQN), to approximate the Q-function, and was the first algorithm to combine deep learning and Q-learning. The network, parameterized by θ, is trained to minimize the loss between the current value estimate $Q(s,a|\theta)$ and the target value $y = \ r + \ \gamma \max_{a\prime\in A} Q(s',a'|\theta^{-})$:

$$\mathcal{L}_Q(\theta) = \ \mathbb{E}_{(s,a,r,s')\sim B}\left[\left(y - Q(s,a|\theta)\right)^2\right], \tag{1.5}$$

where B is a replay buffer of previously observed transitions (s, a, r, s') and θ^- are the network parameters used to compute the target value y. The target network parameters are initially set to θ and only updated towards the new θ every specific number of time steps. This improves learning stability because it mitigates the issue of non-stationary targets, where the target value is computed using the same Q-network being updated.

While deep Q-learning based algorithms allow for learning control policies in large, continuous state spaces, they require discrete actions and are, therefore, not applicable to continuous control problems. In contrast, actor-critic methods, a class of RL algorithms, are well suited for continuous control, since they learn function approximators for both the value function and policy. Actor-critic methods can be divided into two groups of approaches according to how the policy is updated: (i) in parameter space; or (ii) in action space. Here, we will give an overview of Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) and Continuous Actor-Critic Learning Automaton (CACLA) (Van Hasselt and Wiering, 2007; Van Hasselt, 2012) algorithms as examples of the former and latter groups respectively. Both algorithms approximate the policy function using an actor neural network $\mu(\cdot\,|\theta^\mu)$ parametrized by θ^μ. DDPG approximates the Q-function using a critic neural network $Q(\cdot,\cdot\,|\theta^Q)$ parametrized by θ^Q. Similar to Equation 1.5, the critic is trained to minimize the loss between the current value estimate $Q(s_i, a_i|\theta^Q)$ and the target value $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\,\theta^{Q'})$ for a minibatch of n samples:

$$\mathcal{L}_Q(\theta^Q) = \frac{1}{n}\sum\nolimits_i \left[\left(y_i - Q(s_i, a_i|\theta^Q)\right)^2\right], \tag{1.6}$$

where Q' and μ' are the target critic and actor networks parameterized by $\theta^{Q'}$ and $\theta^{\mu'}$ respectively and updated slowly towards their corresponding Q and μ networks:

$$\begin{aligned}\theta^{Q'} &\leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}\\ \theta^{\mu'} &\leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'},\end{aligned} \tag{1.7}$$

with $\tau \ll 1$. The actor is updated by minibatch gradient ascent on the approximate Q-function with respect to the actor parameters:

$$\theta_{t+1}^\mu \leftarrow \theta_t^\mu + \frac{\beta}{n}\sum\nolimits_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i|\theta_t^\mu)}\, \nabla_{\theta_t^\mu}\mu(s|\theta_t^\mu)|_{s=s_i}\,, \tag{1.8}$$

where β is the gradient step size $(0 < \beta \leq 1)$. On the other hand, CACLA approximates the state-value function V using a critic neural network $V(\cdot\,|\theta^V)$ parametrized by θ^V and trained to minimize the loss:

$$\mathcal{L}_V(\theta^V) = \frac{1}{n}\sum\nolimits_i \left[\left(y_i - V(s_i|\theta^V)\right)^2\right], \\ y_i = r_i + \gamma V'(s_{i+1}|\theta^{V'}), \tag{1.9}$$

where V' is the target critic network with parameters $\theta^{V'}$ slowly moving toward their corresponding parameters of the critic network, $\theta^{V'} \leftarrow \tau\theta^V + (1-\tau)\theta^{V'}, with\, \tau \ll 1$. To update the actor, CACLA uses the error in action space between the current estimate of the optimal action $\mu(s|\theta^\mu)$ and an exploratory action a found to produce higher than expected value, as opposed to gradient ascent on the value function used in DDPG. This means that the actor is only updated towards an exploratory action a for which the observed TD error $\delta = r + \gamma V'(s'|\theta^{V'}) - V(s|\theta^V)$ is positive. The reason for this is that when an action results in an increase in the value estimate of a given state ($\delta > 0$), then that action is believed to lead to higher future rewards and is thus made more likely to be selected in that state by driving the output of the actor towards that action. The actor parameters are updated by gradient descent on the loss $\mathbb{E}_{(s,a,r,s')\sim B,\delta>0}\left[\left(a - \mu(s|\theta^\mu)\right)^2\right]$:

$$\theta^\mu_{t+1} \leftarrow \theta^\mu_t + \beta\nabla_{\theta^\mu_t}\, \mathbb{E}_{(s,a,r,s')\sim B,\delta>0}\left[\left(a - \mu\left(s|\theta^\mu_t\right)\right)^2\right], \tag{1.10}$$

where (s, a, r, s') are experience tuples sampled from the replay buffer B and β is the update step size.

1.2.2 Intrinsic Motivation for Reinforcement Learning

Besides the need to learn in continuous action spaces, RL agents need to learn how to explore their environments to collect useful experience data, particularly when extrinsic rewards are delayed and sparsely available. In order to allow RL agents to efficiently and meaningfully explore in a sparse-reward world, intrinsically motivated RL methods have been proposed. These methods endow the agent with the ability to compute an intrinsic reward that models its curiosity to improve its internal knowledge of the world and learn about the consequences of its actions. Different learning measures have been defined to compute an intrinsic reward for the agent, including Bayesian surprise (Schmidhuber, 2010), information-theoretic measures, such as mutual information (Houthooft et al., 2016; Florensa et al., 2017; Barron et al., 2017), entropy of the state distribution (Hazan et al., 2019), and empowerment (Mohamed and Rezende, 2015), variance-based uncertainty (Hester and Stone, 2015), prediction error of a learned world model (Chentanez et al., 2005; Pathak et al., 2017; Shelhamer et al., 2017), predictive learning progress

(Oudeyer et al., 2007; Oudeyer and Kaplan, 2007; Luciw et al., 2011; Gottlieb et al., 2013; Forestier and Oudeyer, 2016), competence progress at achieving self-generated goals (Baranes and Oudeyer, 2013; Forestier et al., 2017; Mannella et al., 2018; Péré et al., 2018), and change in policy value (Özgür and Barto, 2006; Hafez and Loo, 2015).

While most of these measures focus on the perceptual salience and novelty of visited states and, thus, can direct the exploration to states that are highly unpredictable and noisy, learning progress-based measures use the performance improvement in predicting future states over time as well as the novelty of the observed state when deriving an intrinsic reward. This is consistent with recent findings from developmental psychology that show curious exploration in infants to be based on both their own learning history and perceptual variability (Twomey and Westermann, 2015) and with the Goldilocks effect in infant cognition where stimuli of intermediate predictability drive optimal learning with too high or too low predictability being less conductive to learning (Kidd et al., 2012; Kidd et al., 2014).

In addition to improving exploration in tasks with a single goal, intrinsic rewards have been effective in tasks involving a hierarchy or curriculum of goals. For example, intrinsic rewards were used in hierarchical deep RL to encourage a low-level controller to reach a goal state chosen by a high-level controller that learns a policy over intrinsic goals to optimize for an extrinsic reward (Kulkarni et al., 2016). Despite being limited to discrete actions, the proposed algorithm was shown to significantly outperform the DQN algorithm (Mnih et al., 2015) in sparse-reward tasks with a complex goal structure. More recently, intrinsic rewards have been applied in curriculum learning with self-play between two copies of the same agent (Sukhbaatar et al., 2018). The first periodically sets a goal for the second to achieve and is intrinsically rewarded proportionally to the time taken by the second to complete the task, while the second is rewarded inversely proportional to the time taken to complete the task chose by the first, automatically generating a curriculum between themselves. After self-play, an increase in learning speed was shown when the second copy was deployed to solve the target task.

1.3 Summary of Contributions

This section gives an overview of the contributions of this thesis. In Chapter 2, we consider the general problem of learning an exploration strategy in RL. The performance of

RL algorithms strongly relies on how the agent explores its environment to collect experience data to learn from. An undirected exploration behavior limits how fast the agent can learn the desired control policy. Existing approaches mostly involve adding random noise to the current policy, with the risk of spending large amount of valuable training time on exploring parts of the environment the agent has less uncertainty about. Instead, we propose a directed, data-efficient exploration method that integrates self-organization of sensory space with online learning of local predictive models of the environment's dynamics to generate an intrinsic reward for guiding the exploration behavior of the agent. The intrinsic reward is derived from the learning progress in prediction computed locally in each self-organized sensory region using the local model, encouraging the agent to direct its exploration from states of highly predictable sensorimotor dynamics to states of less predictable dynamics. We combine the intrinsic reward with the sparse extrinsic reward to efficiently learn continuous control tasks. Our experiments show that using our exploration method with actor-critic RL leads to significant performance gain in terms of average extrinsic reward and number of actions taken to solve a given task on simple and complex robotic control tasks.

In Chapter 3, we consider the problem of reducing the sample complexity of RL in large-scale domains. Learning control policies from raw high-dimensional observations requires large amounts of training data the agent needs to actively collect online, which is impractical for real-world robot learning. We focus on learning informative low-dimensional state representations to improve the sample efficiency of our intrinsically motivated continuous actor-critic learner on complex visuomotor control learning tasks. We make two main contributions. First, we propose to train the predictive world models as well as the policy in a low-dimensional state space learned unsupervised with hierarchical slow feature analysis (SFA). Our experimental evaluation shows that our SFA-based approach can achieve near-optimal performance on learning vision-based robotic reaching policies in dense and sparse reward environments. Second, we present a novel deep neural network architecture for learning jointly optimized low-dimensional state representations. Our neural architecture is composed of a critic and an actor network. Both networks receive the hidden representation of a deep convolutional autoencoder which is trained to reconstruct the visual input, while the hidden representation is also optimized to predict the value of the state. This state representation captures task-relevant information sufficient to reconstruct the original input and identify rewarding states, since it is learned by jointly optimizing the combined reconstruction and value prediction loss. We show that training a small actor network on this task-relevant low-dimensional state representations allows the policy to be trained with maximum efficiency. The predictive models used to derive the intrinsic reward are trained in the learned representation space. We call the resulting algorithm deep intrinsically motivated continuous actor-critic (Deep ICAC). Our simulation and real-world robotic experiments show state-of-the-art performance of our Deep ICAC on learning-to-reach and learning-to-grasp tasks from raw visual input in different reward settings.

In Chapter 4, we introduce the Curious Meta-Controller (CMC), a novel approach for arbitrating between model-based and model-free control in deep RL. The arbitration is adaptively determined based on the model reliability estimated by the learning progress in predictions. The learning progress is used to derive an intrinsic reward representing the agent's curiosity to seek data that improves its model of the world. When the model is found to be sufficiently reliable during arbitration, the model-based system performs gradient-based action optimization and the optimal action is used as a more informed exploratory action for the model-free learner. Unlike previous works, our CMC approach considers the reliability of the model when deciding on which of the model-based and model-free control systems to query for an action at each time step and does not require a predefined threshold to arbitrate between them. CMC can be combined with any off-policy actor-critic method. We demonstrate that using CMC for meta-decision making leads to fast and stable performance on learning robotic reaching and grasping tasks from raw-pixel input.

In Chapter 5, we consider the problem of experience imagination in deep RL. A predictive model of the world can generally be leveraged to provide additional experience data for training the RL agent without the need for expensive environmental interactions. Generating imagined data for visuomotor control tasks, however, requires learning perfect world models at the pixel level, which is infeasible in practice. We propose a novel learning-adaptive imagination approach that performs experience imagination in a learned latent space. In our approach, the learned latent space is self-organized into local regions with local world models, and a running average of model prediction error is independently computed for each region. The experience replay buffer is divided into pixel-space and latent-space buffers for storing real and imagined experiences respectively. Imagined rollouts are reliably generated with probability inversely proportional to the average error of the current region, and the imagination depth is adaptively determined by the average error of the traversed regions. To encourage collecting data that improves future predictions necessary for imagination, we use an intrinsic reward based on the spatially and temporally local learning progress. Our experiments show that our approach to imagination makes learning pixel-level control policies for robotic grasping more efficient, particularly in sparse reward environments.

In Chapter 6, we present a novel robot dual-system motor learning approach that is behaviorally and neurally plausible, data-efficient, and competitive with the state of the art. Our approach improves on the previously proposed curious meta-controller (CMC) approach by enabling an adaptive-length model rollout for plan optimization during model-based control. This is done by incrementally self-organizing the space of latent state representations and computing the reliability estimate locally for every region of the learned latent space. Rolling out the model until the estimated reliability is low, as opposed to using a fixed time horizon, ensures that no imperfect model predictions are used in computing the optimal plan and reduces the computational cost. Our approach also integrates online arbitration with offline experience imagination into a single learning framework where imagined experiences collected from model rollouts are used as additional training data for the control policy. We show that our approach learns better vision-based control policies than baseline and state-of-the-art methods in dense and sparse reward environments. Policy networks trained in simulation with our approach are shown to perform well on the physical robot without fine-tuning of the trained policies.

1.4 Thesis Organization

This section gives an overview of the following chapters that comprise the body of this thesis. Chapter 2 presents a directed, data-efficient exploration strategy inspired by sensorimotor development in infants. It guides the robot’s choice of action by providing an intrinsic reward based on the rate of progress in learning a predictive world model, informatively shifting the interactions from well-explored to less-explored regions of the world. In Chapter 3, we study the effect of state representation learning on the performance of RL for robot motor control from raw visual input and describe a new neural network architecture for learning low-dimensional, task-relevant state representations.

The low-dimensional representation, which is jointly trained to optimize the supervised value prediction and unsupervised state reconstruction objectives, is used as a direct input to the policy and the world models whose prediction errors are used in deriving the intrinsic reward, effectively improving the sample efficiency. Chapter 4 discusses different hypotheses on how the brain arbitrates between model-based and model-free learning systems and shows how the learning progress-based intrinsic reward can be used as an estimate of model reliability to adaptively decide when to switch from model-free to model-based control and vice versa during robot motor learning. Chapter 5 establishes a bridge between intrinsic motivation and experience imagination in robot decision-making, inspired by human mental simulation of motor behavior. The proposed approach augments the training set of real experiences with imagined experiences generated with a learned model over regions where the model is reliable and uses the intrinsic motivation to collect experience data that improves the model. In Chapter 6, we propose a new dual-system motor learning approach that integrates arbitration with imagination and enables an adaptive-length model rollout for plan optimization during model-based control and for generating imagined experiences. Finally, Chapter 7 concludes by summarizing the thesis and discussing potential avenues for future research.

Chapter 2

Spatially and Temporally Local Learning Progress

2.1 Introduction

The performance of RL algorithms strongly relies on the chosen exploration strategy, as it is the only way to discover more useful experience data and eventually better control policies. Existing approaches are mostly based on adding random noise to the policy output. While this undirected exploration might work in simple domains, it severely limits how quickly the algorithm can converge to a promising policy in domains with continuous state-action spaces.

Instead of the commonly used random exploration strategies, count-based exploration approaches apply the principle of optimism in the face of uncertainty by storing the number of times a state-action pair has been visited and assign a value inversely proportional to the state-action visitation frequency to encourage exploring parts of the state-action space that are less familiar (Brafman and Tennenholtz., 2002; Lihong et al., 2009). More recently, extending count-based exploration, which is limited to tabular representations of the environment, to problems with continuous high-dimensional state spaces has been proposed (Fu et al., 2017; Tang et al., 2017; Ostrovski et al., 2017; Machado et al., 2019). In (Stadie et al., 2015), an exploration bonus based on the prediction error of a learned dynamics model is added to the reward to encourage visiting novel states. Other approaches use an estimate of the uncertainty in the Q-values to improve exploration. For example, one approach approximates a distribution over Q-functions by learning K bootstrapped Q-value estimates using a deep neural network with shared body and K heads corresponding to the bootstrapped estimates (Osband et al., 2016). A bootstrapping binary mask of length K is sampled and attached to each collected experience to determine which of the K heads the experience will be used to train during experience replay. Exploration is performed by uniformly sampling a Q-function head at the start of each episode and following the action with the highest estimated Q-value. In (Moerland et al., 2017), a Gaussian distribution over returns is learned by a neural network that outputs a mean and standard deviation of the Q-value estimates and is used to represent return uncertainty. Both the return uncertainty and the uncertainty over the neural network parameters are propagated through the Bellman equation when updating value estimates. Exploratory actions are then directly chosen by Thompson sampling.

While these approaches improve the exploration in complex environments with continuous state spaces, they are typically limited to discrete action spaces and cannot be applied to continuous-control robotic tasks. An alternative and behaviorally plausible line of work is to use intrinsic motivation to induce directed exploration behavior. Intrinsic motivation methods build an internal representation of how the world evolves over time as a function of the agent's actions and derive an exploration behavior that improves the representation. This is mostly done by learning a predictive model of the world that predicts the next state given the current state and action and encouraging actions that lead to data that improves model predations. Intrinsically motivated exploration is particularly effective in situations where agents lack the important feedback on how to adjust their behavior during task learning, such as learning in environments with sparse or delayed extrinsic rewards.

Most learning measures used to generate intrinsic rewards are based on the prediction error of the world model. However, simply using an error in predicting future states to derive an intrinsic reward is very likely to direct exploration towards noisy states that retain constant, large prediction error. Learning progress-based measures (Oudeyer et al., 2007; Luciw et al., 2011; Gottlieb et al., 2013), on the other hand, use the change in model prediction error over time when computing the intrinsic reward and therefore avoid focusing the exploration on regions of inherently unpredictable dynamics. These approaches usually learn a single global model of the world and 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, they only inform about the prediction performance over the entire sensory space, which is difficult to cover, and assume correct generalization of the learning progress from regions where recent experience samples have been collected to novel regions. This generalization cannot be guaranteed since the learning progress is locally non-stationary (prediction errors keep changing as the model evolves). Moreover, learning a perfect global model requires large amounts of training samples that go beyond the agent's lifetime.

In this chapter, we propose a directed, data-efficient exploration approach. Our approach learns a growing ensemble of local world models by self-organizing the sensory space into local regions and training a separate model locally in each region. We use the time derivative of the average prediction error of the model over a window of recent predictions in each region to define spatially and temporally local learning progress. The local learning progress is then used to derive a noise-robust intrinsic reward, which is combined with the extrinsic reward to guide exploration. We integrate our directed exploration with an actor-critic architecture to learn continuous control policies more efficiently and call the resulting algorithm Intrinsically motivated Continuous Actor-Critic (ICAC). Guiding exploration via spatially and temporally local learning progress resembles how infants continually organize their interaction with the world as they learn about its dynamics, shifting their focus from well-explored to less-explored regions driven by their curiosity. Our learning progress-based intrinsic reward accords with surprise-enhanced learning, where violation-of-expectation, quantified as the difference between the predicted and the observed stimulus, enhances children's learning (Stahl and Feigenson, 2017), and the Goldilocks effect principle in infant cognition that states that stimuli of intermediate predictability are essential for optimal learning, as opposed to highly predictable or unpredictable stimuli (Kidd et al., 2012). Our experiments show significant performance gain of our ICAC algorithm, in terms of average extrinsic reward and number of actions taken to solve a given task, compared to the actor-critic baseline on simple and complex robotic control tasks in sparse reward environment.

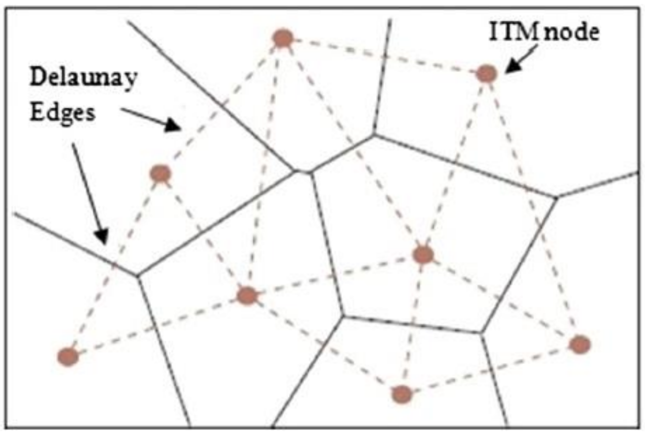


Figure 2.1: ITM network. The nodes of the incrementally built ITM network are the centers of the Voronoi cells. The Delaunay triangulation (dotted lines) connects the centers of the neighboring cells such that no center lies inside the circumcircle of any triangle.

2.2 Growing Self-Organization of Sensory Space

Similar to how an infant progressively learns about its environment, we want our learning agent to organize its interaction with the environment by changing the focus of exploration from regions where it has learned about the outcome of its motor actions to regions where it expects to learn new effects of motor activity. To facilitate this, the sensory space is incrementally partitioned into local regions with local world models by using a growing self-organizing network. Particularly, we use the Instantaneous Topological Map (ITM) network model (Jockusch and Ritter, 1999). This is because ITM is originally designed for strongly correlated stimuli, which is the case here since the stimuli are generated by exploring the sensory space along continuous trajectories. ITM is an unsupervised learning method for adaptively building a topology-preserving map of an input space by constructing the Delaunay triangulation of the generated nodes in the map (Figure 2.1). Unlike other common self-organizing maps such as the SOM (Kohonen, 1989), Neural Gas (Martinetz and Schulten, 1991), and Growing Neural Gas (Fritzke, 1995), ITM has fewer hyperparameters and is considered more computationally efficient, with the number of nodes scaling linearly with the volume of the sensory space.

The ITM network is defined by a set of nodes i, each having a weight vector w_i, and a set of edges connecting each node i to its neighbors $N(i)$. The network starts with two connected nodes, and when a new stimulus s is observed, the following adaptation steps are performed:

1. Matching: Find the nearest node n and the second-nearest node n' to s: $n \leftarrow \arg\min_{i} \|s - w_i\|_2^2$, $n' \leftarrow \arg\min_{j, j \neq n} \|s - w_j\|_2^2$.
2. Edge adaptation: If n and n' are not connected, add an edge between them. For all nodes $m \in N(n)$, if n' lies inside the Thales sphere through *m* and *n* (the sphere with diameter $w_m w_n$), $i.e. (w_n - w_{n'}) \cdot (w_m - w_{n'}) < 0$, remove the edge between *m* and *n* (non-Delaunay edge), and if *m* has no remaining edges, remove *m*.
3. Node adaptation: If s lies outside the Thales sphere through *n* and n', $i.e. (w_n - s) \cdot (w_{n'} - s) > 0$, and if $\|s - w_n\|_2^2 > e_{max}$, where e_{max} is the desired mapping resolution, create a new node v with $w_v = s$ and an edge with *n*.

Each region n of the sensory space (node in ITM) is assigned a local world model $\mathcal{M}_n(\cdot,\cdot\,|\theta^{\mathcal{M}_n})$ approximated by a neural network with parameters $\theta^{\mathcal{M}_n}$. The model is trained online from experience data to predict the next state, given the current state and action. When observing a new state s_t, the nearest node *n* of ITM to s_t is determined and the associated model $\mathcal{M}_n$ is updated by minimizing the loss $\left\|\mathcal{M}_n(s_t, a_t|\theta^{\mathcal{M}_n}) - s_{t+1}\right\|_2^2$ between the true and the predicted next state, s_{t+1} and $\mathcal{M}_n(s_t, a_t|\theta^{\mathcal{M}_n})$ respectively.

In the following section, we describe how the self-organized sensory space and the local world models can be used to derive spatially and temporally local learning progress and present our ICAC algorithm that generates an intrinsic reward based on the derived learning progress to direct exploration.

2.3 Intrinsically Motivated Continuous Actor-Critic

During task learning, a moving average of the model prediction error over a window of σ recent predictions is computed and updated separately for each region *n* of the state space (node in ITM):

$$\langle e_{t,n}^{prd} \rangle = \frac{1}{\sigma} \sum_{i=1}^{\sigma} e_i^{prd} \,\Big|_{e_i^{prd} = \|\mathcal{M}_n(s_i, a_i|\theta^{\mathcal{M}_n}) - s_{i+1}\|_2^2}, \tag{2.1}$$

where e_i^{prd} is the prediction error related to the state transition (s_i, a_i, s_{i+1}). The change in average prediction error $\langle e_{t,n}^{prd} \rangle$ over time represents the learning progress (LP) the agent has made or expects to make and is then estimated using a time window $\mathcal{W}$:

$$LP_{t,n} = \left|\langle e_{t-\mathcal{W},n}^{prd} \rangle - \langle e_{t,n}^{prd} \rangle\right|. \tag{2.2}$$

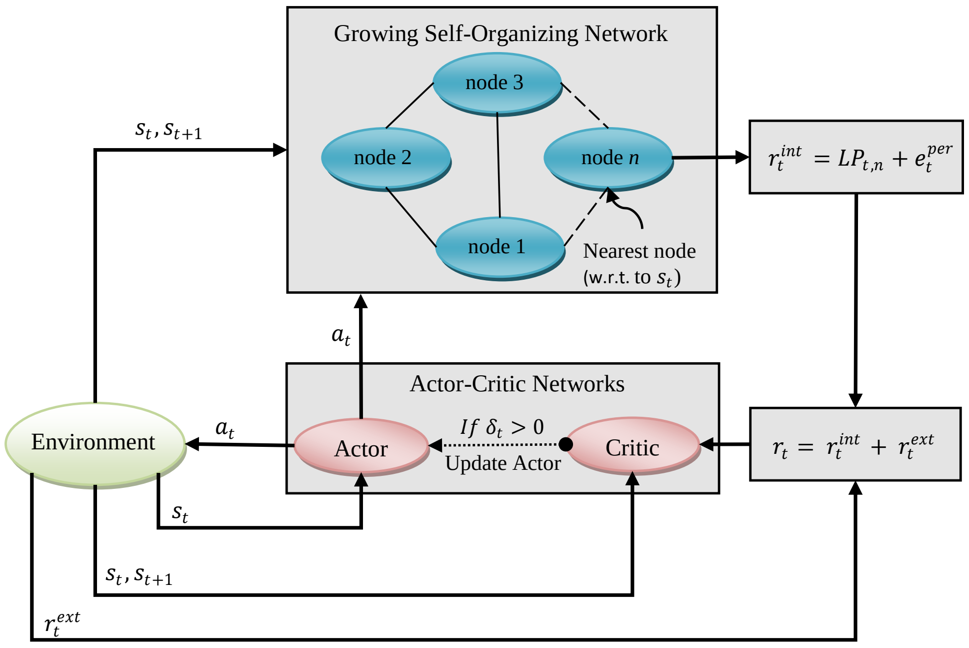


Figure 2.2: ICAC learning system: The growing self-organizing network adaptively forms a topology-representing map of the agent's sensory space, where each node is associated with a local world model capturing the dynamics in the region covered by the node. The world model associated with the nearest node to the observed state s_t predicts the next state using both s_t and the action a_t and is updated using the true next state s_{t+1}. An intrinsic reward is then generated from the perception error and the learning progress observed, combined with the extrinsic reward and sent to the critic to update its estimate of the utility of a_t. The actor is updated towards a_t, if the resulting TD error (δ_t) is positive.

When action a_t is taken at state s_t, the resulting next state s_{t+1} is used to compute e_t^{prd} and then $\langle e_{t,n}^{prd} \rangle$ and $LP_{t,n}$ associated with the best-matching node n of the ITM w.r.t. s_t are updated. The updated $LP_{t,n}$ is combined with the perception error $e_t^{per} = \|s_{t+1} - w_m\|_2^2$ which is the distance between the state s_{t+1} and the weight vector of the nearest node m to s_{t+1}, to give an intrinsic reward signal:

$$r_t^{int} = LP_{t,n} + e_t^{per} \,. \tag{2.3}$$

This intrinsic reward models the agent's curiosity by directing its exploration towards areas that are perceptually novel and where maximum learning progress is expected. It facilitates moving from well-explored, where no learning progress is expected, to less-explored regions of the sensory space, where large learning progress is expected. By relying on the time derivative of the average prediction error instead of the average error itself, the intrinsic reward ensures that the agent is not attracted to states where it always gets inevitably high prediction error, due to noisy or non-deterministic dynamics. The intrinsic reward is then combined with the extrinsic reward:

$$r_t = r_t^{int} + r_t^{ext}. \tag{2.4}$$

The combined reward r_t is then passed to the critic to update its estimate of the value of s_t. Here, we use CACLA (Van Hasselt, 2012) as the base actor-critic algorithm. The two neural networks representing the actor and critic are $\mu(s|\theta^{\mu})$ and $V(s|\theta^{V})$ parametrized by θ^{μ} and θ^{V} respectively. As discussed in Section 1.2.1, CACLA updates its actor towards an exploratory action that has been found to produce higher than expected value, which happens when the observed TD error (δ) is positive. To emphasize the effect of actions yielding better improvements to the expected value than usual, CACLA can further magnify the actor update. This is done by first keeping a running average of the TD error's variance $var_t = (1-\beta)\, var_{t-1} + \beta {\delta_t}^2$, with $\beta \ll 1$, and then the number of optimization steps performed for updating towards an action is determined by $\lceil \delta_t / \sqrt{var_t} \rceil$, which shows how many standard deviations the obtained value is above the expected value, rounded up to the next nearest integer number. This is referred to as CACLA+Var to distinguish it from CACLA where at most a single optimization step is performed per update. CACLA is on-policy and the probability of selecting action a at state s is given by the chosen policy distribution, e.g. Gaussian centered at $\mu(s|\theta^{\mu})$ with standard deviation σ: $\pi(a|s) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(a - \mu(s|\theta^{\mu})\right)^2 / 2\sigma^2}$ (see Appendix B). The learning system, including the actor-critic architecture and the growing self-organizing network, is shown in Figure 2.2. The proposed ICAC algorithm is detailed in Algorithm 1.

2.4 Experiments

In the following, we describe the experimental setup and present two robotic experiments for learning increasingly difficult control tasks. All parameter values were empirically determined after preliminary testing on the environments considered. The experiments were run using a discount factor γ 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 σ 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

Algorithm 1 Intrinsically motivated Continuous Actor-Critic (ICAC)

```
1:  Initialize actor and critic network parameters $\theta^{\mu}$ and $\theta^{V}$
2:  Initialize the ITM network
3:  for $episode = 1 \; to \; E$ do
4:      Sample initial state $s_1$
5:      for $t = 1 \; to \; T$ do
6:          Sample action $a_t \sim \pi$: $\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\left(a_t - \mu(s_t|\theta^{\mu})\right)^2/2\sigma^2}$
7:          Execute $a_t$ and observe $r_t^{ext}$ and $s_{t+1}$
8:          Update ITM and the local world model of the region covering $s_t$
9:          Compute intrinsic reward $r_t^{int}$ using Equation 2.3
10:         Compute total reward $r_t$ using Equation 2.4
11:         Update $\theta^{V}$ to minimize the loss $\frac{1}{2}\left(y - V(s_t|\theta^{V})\right)^2$, where $y = r_t + \gamma V(s_{t+1}|\theta^{V})$
12:         Compute TD error $\delta_t \leftarrow y - V(s_t|\theta^{V})$
13:         Update the variance of TD error $Var_{t+1} \leftarrow (1-\beta)Var_t + \beta{\delta_t}^2$
14:         if $\delta_t > 0$ then
15:             Update $\theta^{\mu}$ to minimize the loss $\frac{1}{2}\left(a_t - \mu\left(s_t|\theta_t^{\mu}\right)\right)^2$ by performing $\left\lceil \frac{\delta_t}{\sqrt{var_t}} \right\rceil$ optimization steps.
16:         end if
17:     end for
18: end for
```

activation functions for the hidden and output layers respectively. The networks were trained using stochastic gradient descent with a learning rate of 0.01. To compute the running average of the variance of the TD error, we used a learning rate β of 0.01. Varying β did not affect the performance adversely.

The desired mapping resolution e_{max} of the ITM network was set to 0.9. Smaller values were found to increase the computation time without considerable performance gain. All local world models 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 time windows σ and $\mathcal{W}$ used in computing the local learning progress were set to 40 and 20 time units respectively. All inputs to the neural networks were normalized to the interval $[-1, 1]$.

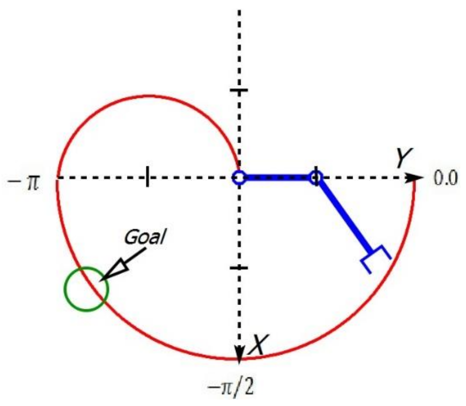


Figure 2.3: Goal reaching with 2-DoF robotic arm. The red curve outlines the reachable workspace for the arm. The green circle is the current goal zone.

2.4.1 Reaching with 2-DoF Robotic Arm

In this experiment, we test our ICAC algorithm on a simple control task of reaching a random goal in 2D space with a 2-DoF robotic arm, as shown in Figure 2.3. The state representation used in the actor and critic networks is 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:

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

Both links of the arm are 1 unit length and the goal zone radius is 0.3 unit length.

Learning is performed over 1000 episodes, and the agent is given a maximum of 50 time steps 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 random seeds. We compare our ICAC algorithm to CACLA and CACLA+Var. Figure 2.4 shows the average extrinsic reward obtained per episode of the three algorithms. Although the extrinsic reward was sparse, being received 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, Figure 2.5 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 just below 10 actions on average toward the goal. The reason CACLA does not quickly find an optimal

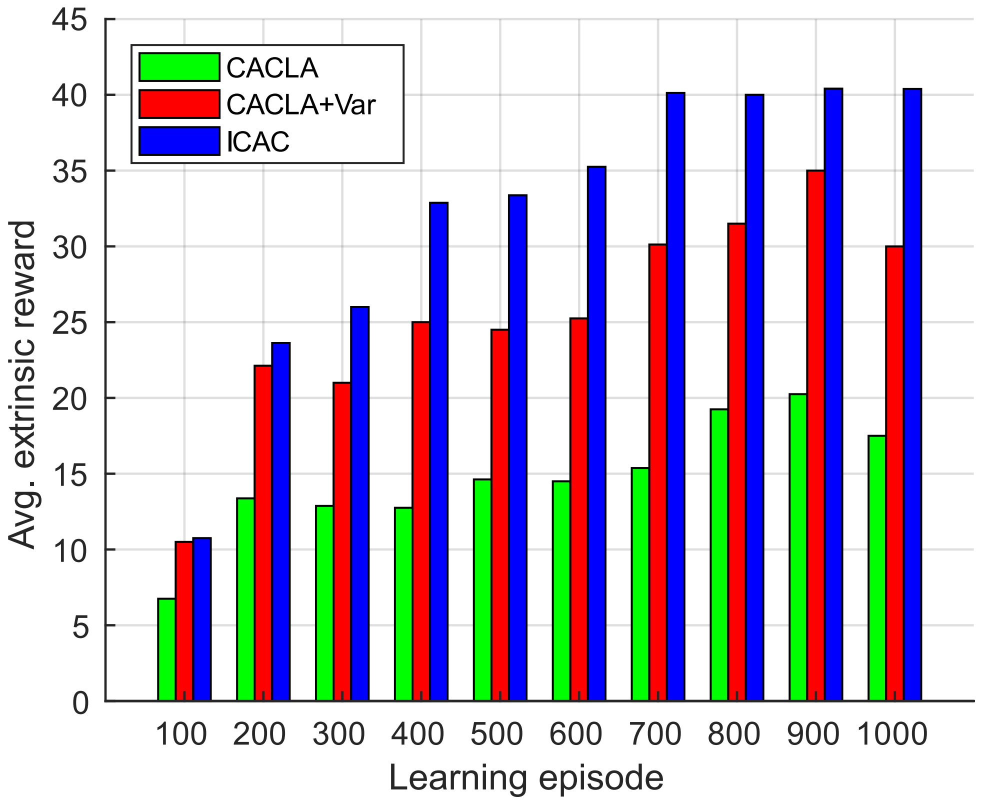


Figure 2.4: Average extrinsic reward on random goal reaching with 2-DoF robotic arm.

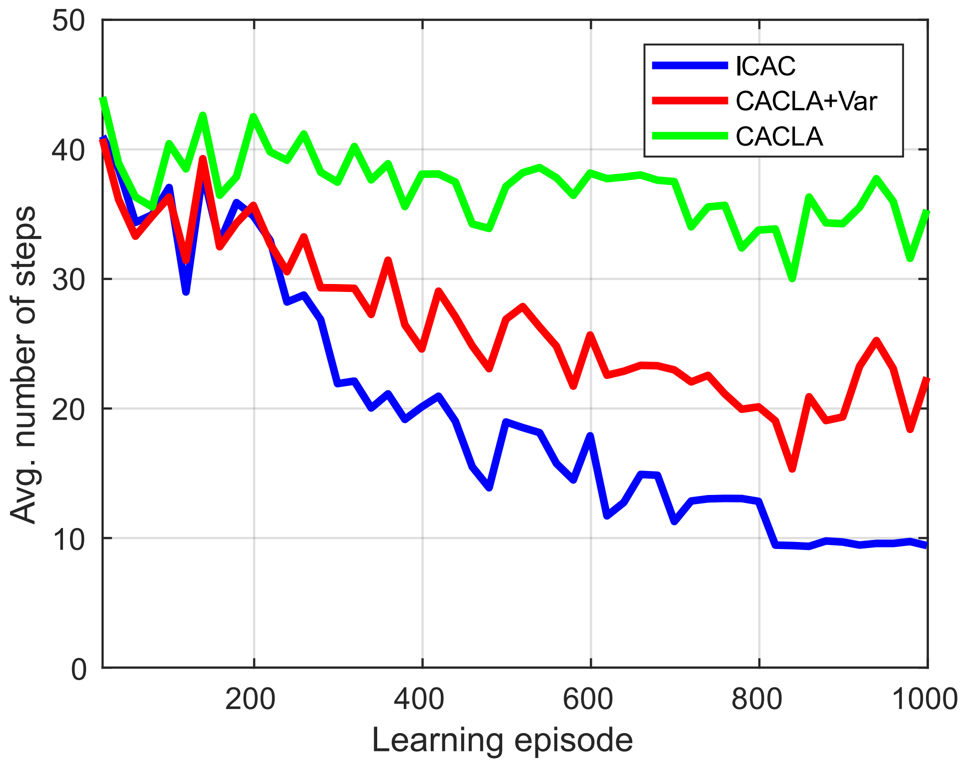


Figure 2.5: Average number of steps to the goal on random goal reaching with a 2-DoF robotic arm. The average over 20 episodes is shown for readability.

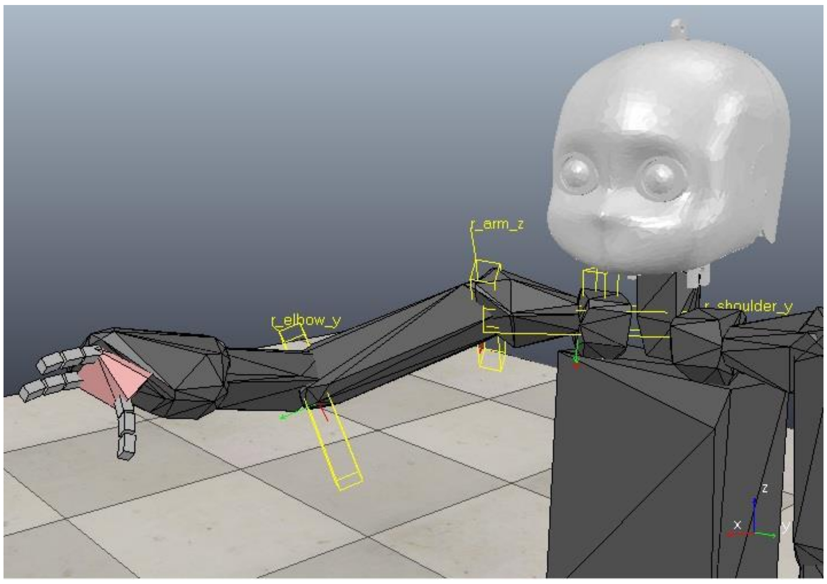


Figure 2.6: Simulated NICO robot. Four joints in the right arm are used to learn to perform a reaching task in the 3D workspace.

policy, as seen in Figure 2.4 and Figure 2.5, 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 chooses actions that maximize the learning progress. These actions keep changing as the perception error and the local world models evolve, allowing the discovery of new policies with higher rewards and fewer actions.

2.4.2 Reaching with 4-DoF NICO Arm

We evaluate here the three algorithms on our Neuro-Inspired COmpanion (NICO) robot (Kerzel et al., 2017) in 3D space. The task is to learn to move 4-DoF robotic arm to reach the desired goal region. The experiment was run in the V-REP robot simulator (Rohmer et al., 2013), as shown in Figure 2.6. The joints considered in the experiment are shown in Table 2.1. 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 is 7-dimensional real-valued vector consisting of the 4D state representation and the Cartesian coordinates of the current goal position. 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. No extrinsic rewards are provided until the robot reaches the goal region in which it receives a positive extrinsic reward of 100:

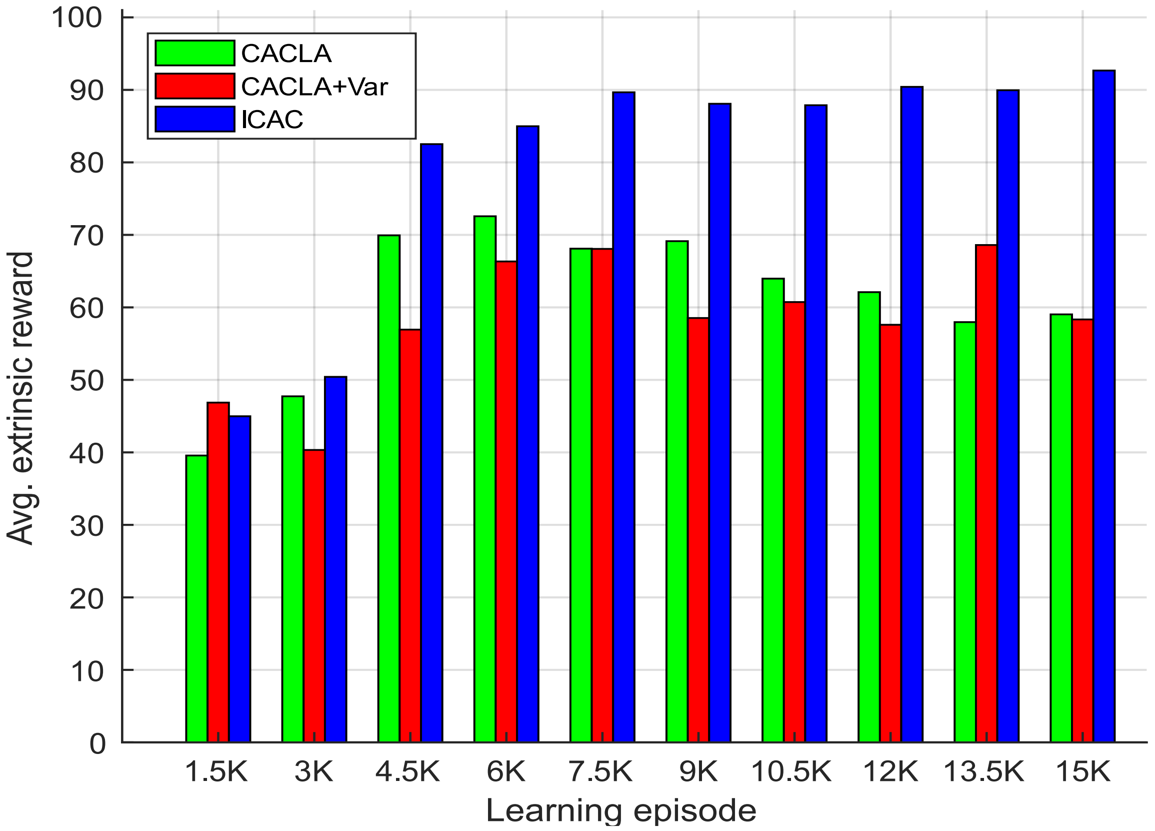


Figure 2.7: Average extrinsic reward on random goal reaching with 4-DoF NICO arm.

Table 2.1: The joints considered in the reaching experiment with 4-DoF NICO arm.

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.	$[-100, 100]$

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

Each simulation experiment consists of 15K 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 random seeds. The average extrinsic reward obtained by the robot running each of the three algorithms is shown in Figure 2.7. CACLA and CACLA+Var algorithms were able to reach

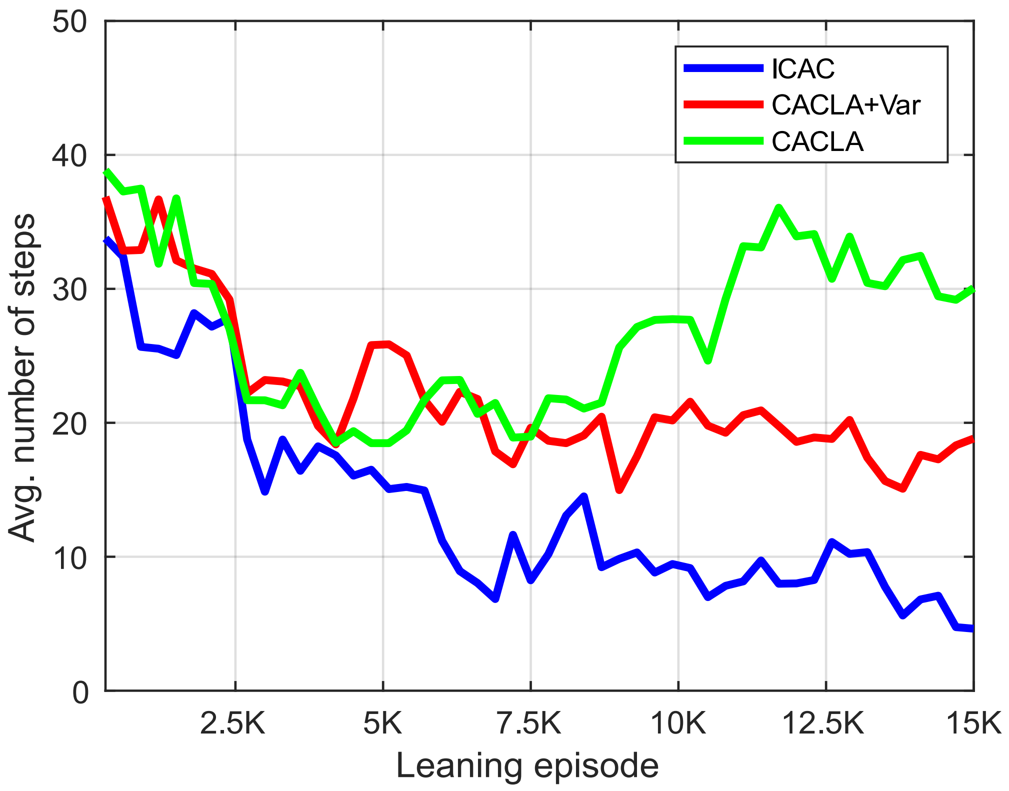


Figure 2.8: Average number of steps to the goal on random goal reaching with 4-DoF NICO arm. The average over 300 episodes is shown for readability.

only a maximum average reward of around 60 after 4.5K episodes, indicating that they failed to reach the goal in 8 out of 20 simulation runs (60% success). 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. Regarding the average number of steps to the goal, CACLA and CACLA+Var learned a policy of 30 and 20 steps respectively, whereas 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 Figure 2.8.

2.5 Conclusion

In this chapter, we presented a directed exploration approach for continuous-action RL that is data-efficient, noise-robust, and compatible with infant sensorimotor development and proposed the ICAC algorithm that integrates our directed exploration with continuous actor-critic RL architecture. In our approach, the sensory space is self-organized into local regions with local world models. The change in average prediction error of a learned local model over time defines spatially and temporally local learning progress, which is used to derive an intrinsic reward to direct exploration from highly to less predictable areas of the sensory space. The derived intrinsic reward is combined with the extrinsic reward and fed as input to the critic of our ICAC algorithm to update its estimate of the state value. The actor is updated towards actions found to produce higher than expected value. We evaluated our approach on increasingly difficult robotic control tasks in sparse reward environments. Our experimental results show that our ICAC algorithm can learn goal-reaching policies substantially faster and with more stability than two continuous actor-critic baselines. The results also show that ICAC converges to better action policies, in terms of average extrinsic reward and average number of steps taken to reach a random goal, than the baselines. Overall, the results indicate that our local learning progress-based intrinsic motivation is effective in acquiring sensorimotor skills in sparse reward environments. This suggests a potentially important role for our approach in open-ended developmental learning systems where complex behavior incrementally develop without heavily relying on the extrinsic feedback.

Chapter 3

Intrinsically Motivated Actor-Critic with Efficient State Representations

3.1 Introduction

The ability to map raw and high-dimensional sensory input to motor output that optimizes the task performance is essential for the acquisition of visuomotor skills in robots. Deep RL enables learning a policy function representing that mapping from trial and error experience in the environment, utilizing deep neural networks as function approximators. However, deep RL suffers from poor sample efficiency as it requires large amounts of self-collected training data to adjust the large number of learning parameters such deep architectures typically have, rendering it generally impractical for real-world robotic learning.

To improve sample efficiency in deep RL, different approaches have recently been proposed. Some focused on improving experience replay in which data are drawn from a memory of recent experiences and used for training. Sampling experiences with probability proportional to reward prediction error instead of random sampling (Schaul et al., 2016) and counting unsuccessful policy rollouts as successful ones by replaying experiences with a different goal than the one the agent was trying to achieve (Andrychowicz et al., 2017; Levy et al., 2019) are two prominent examples. Another approach proposed learning an estimate of the expected future state occupancy from any given state-action pair, called the successor representation (SR) (Kulkarni et al., 2016). This allowed for replacing the state-action value function, which estimates the expected long-term reward, with a function estimating only the immediate reward using SR, and thereby eliminating the need for the slow propagation of state-action values among visited states. In (Jaderberg et al., 2017), it was shown that maximizing auxiliary rewards that are proportional to the perceptual changes in the raw observation as well as the learned feature spaces while learning the target task makes the training faster. To achieve this, auxiliary policies were learned that maximize the perceived changes in pixel values of input images and in neuron activations of each layer of the value and policy networks, optimizing the combined loss of the auxiliary and the base tasks. Learning a separate exploration policy was also found to increase the sample efficiency of learning the target policy by updating the exploration policy based on the amount of improvement in the target policy as a result of the experience data collected with the exploration policy (Xu et al., 2018). Similarly, the learning speed on novel tasks was found to be improved by using a task-independent exploration policy updated between learning trials of different tasks (Garcia and Thomas, 2019).

While these approaches offer a variety of techniques in which sample efficiency in deep RL can be improved, training state representations are limited to labeled data that is slowly collected and not always available, particularly when rewards are delayed or sparse. This prevents the convergence to near-optimal policies in a reasonable amount of time. In tasks that require learning control from high-dimensional sensory observations, such as raw images, it is helpful for the agent to learn low-dimensional abstract representations in an unsupervised manner to increase the speed of learning. State representation learning in RL has received considerable attention over the past decade. For instance, using autoencoders to learn compact low-dimensional state representations unsupervised for RL has been proposed (Lange and Riedmiller, 2010; Lange et al., 2012; Finn et al., 2016). However, a common limitation to these methods is that they require a separate pre-training phase to adjust the autoencoder weights prior to learning the policy for the target task. Therefore, they learn features that are not relevant to the task and do not necessarily identify rewarding states, leading to poor task performance.

In this chapter, we address the sample complexity of learning continuous control from raw visual input by proposing two approaches corresponding to two different types of unsupervised learning of abstract state representations. In the first approach, we use low-dimensional state representations learned unsupervised with hierarchical slow feature analysis to train the actor and critic networks. Slow feature analysis (SFA) is an unsupervised learning method that learns compact and temporally coherent features from an input sequence of observations (Wiskott and Sejnowski, 2002). The spatially and temporally local learning progress is used to derive an intrinsic reward for guiding exploration. The local world models used in computing the learning progress are trained on the slow features. We evaluate the approach on learning robotic reaching from raw-pixel data in a realistic robot simulator. The results show that the control policies learned with our approach are better both in terms of length and average extrinsic reward than the actor-critic baseline. In the second approach, we propose a jointly trained deep neural architecture composed of a convolutional autoencoder, actor, and critic networks. The hidden representation of the autoencoder is trained to reconstruct the visual input and predict the expected state value. This low-dimensional representation captures the important task-relevant information, since it encodes features necessary to discriminate the state and predict its value by minimizing the joint unsupervised reconstruction and supervised value prediction loss. Therefore, it is used as an input to a small actor network to enable faster learning of the target policy. To efficiently direct the exploration of the actor-critic learner, we use an intrinsic reward based on the spatially and temporally local learning progress in the learned representation space. The resulting algorithm is called Deep Intrinsically motivated Continuous Actor-Critic (Deep ICAC). Our experiments show that the control policies learned with our Deep ICAC algorithm can achieve better performance than the compared state-of-the-art and baseline algorithms on vision-based learning-to-reach and learning-to-grasp tasks in simulation and in the real world.

3.2 ICAC with Slow Features

Slow feature analysis (Wiskott and Sejnowski, 2002) as an unsupervised method for learning slowly varying features from an input signal has been successfully applied to provide a low-dimensional feature vector for precise prediction of view-invariant attributes of objects, such as identity, position and orientation in the context of supervised learning (Franzius et al., 2011), for simultaneous learning of place and head-direction cells for potential robot navigation (Zhou et al., 2017), for human action recognition

(Zhang and Tao, 2012; Yousefi and Loo, 2016), and for detecting changes in facial expression from video sequences (Zafeiriou et al., 2013).

In this section, we introduce our approach for learning vision-based continuous control polices with our local learning progress-based intrinsic motivation and using state representations trained unsupervised with slow feature analysis (SFA). We first provide a background on SFA and describe the architecture and training process of a hierarchical SFA network. We then show how the output of a trained SFA network can be used as a low-dimensional state representation in our Intrinsically motivated Continuous Actor-Critic (ICAC) algorithm.

3.2.1 Hierarchical Slow Feature Analysis

The learning problem in SFA is an optimization problem of finding the most slowly varying features in an input signal. Given an I-dimensional input signal $x(t)$, the task is to find a set of J input-output functions $g(x) = [g_1(x), \dots, g_J(x)]^T$ such that the output signals $y_j(t) = g_j(x(t))$ minimize

$$\Delta(y_j) \coloneqq \langle \dot{y}_j^2 \rangle \tag{3.1}$$

under the constraints

$$\langle y_j \rangle = 0 \qquad (zero\ mean), \tag{3.2}$$

$$\langle y_j^2 \rangle = 1 \qquad (unit\ variance), \tag{3.3}$$

$$\forall\, i < j : \langle y_i y_j \rangle = 0 \qquad (decorrelation) \tag{3.4}$$

with $\langle . \rangle$ and $\dot{y}$ indicating temporal averaging and the time derivative of y respectively. Constraints (3.2) and (3.3) avoid a trivial solution of constant output and constraint (3.4) ensures that different functions g_j contribute different features. This is a variational calculus optimization problem and is generally difficult to solve. If the functions g_j are constrained to be linear combinations of nonlinear functions, then the problem is simplified. To obtain the optimal solution, SFA searches for the direction along which the time derivative of the input signal varies most slowly by performing the following steps:

1. Nonlinear expansion: The input signal x is normalized to zero mean and unit variance and then nonlinearly expanded by adding degree one and degree two monomials including all mixed terms, such as $(x_1 x_2)$, using the function $\tilde{h}(x)$. The resulting signal has the following form:

$$\tilde{z}(t) = \tilde{h}(x(t)) = [x_1(t), \ldots, x_I(t), x_I(t)x_1(t), x_I(t)x_2(t), \ldots, x_I(t)x_I(t)]^T.$$

2. Sphering: The expanded signal $\tilde{z}$ is normalized to zero mean and unit covariance. This normalization is called sphering since it ensures the components of the signal are uncorrelated and have the same variance. Sphering, referred to here by function $h(\mathrm{x})$, is done by applying principal component analyses (PCA) on the covariance matrix $\langle(\tilde{z} - \langle\tilde{z}\rangle)(\tilde{z} - \langle\tilde{z}\rangle)^T\rangle$, resulting in the sphered signal $z,\ z = h(\mathrm{x})$.
3. Principal component analyses: PCA is applied on the covariance matrix $\langle\dot{z}\,\dot{z}^T\rangle$ to find the *J* eigenvectors v_j that correspond to the *J* smallest eigenvalues

$$\begin{aligned} v_j: \quad \langle\dot{z}\,\dot{z}^T\rangle v_j \ &= \ \lambda_j v_j \\ \text{with } \lambda_1 \leq \lambda_2 &\leq \ldots \leq \lambda_J. \end{aligned} \tag{3.5}$$

The eigenvectors v_j are the directions of the smallest variations of the time derivative signal $\dot{z}$ and specify the input-output functions $g(x) = [g_1(x), \ldots, g_J(x)]^T$ as follows:

$$g_j(\mathrm{x}) = v_j{}^T h(\mathrm{x}). \tag{3.6}$$

To extract the *J* slowest features from any new signal $\acute{x}$, the function $g(\acute{x})$ is computed, as shown in Equation 3.6, which projects the signal vectors onto the eigenvectors v_j.

In our approach, we apply SFA hierarchically. This is both computationally efficient and biologically realistic since hierarchical ordering requires only local communications similarly to how extensive connectivity is avoided in the neural circuits of the visual cortex (Koulakov and Chklovskii, 2001). Our hierarchical architecture consists of three layers of SFA nodes (see Figure 3.1). Each node in the first layer has a receptive field of 8×8 pixels in the original 64×64 pixel input with 4-pixel overlap, resulting in 15×15 nodes with partially overlapping receptive fields. The second layer's nodes have receptive fields of 6×6 nodes in the lower first layer with 3-pixel overlap, resulting in 4×4 nodes. The top layer has a single SFA node that covers the full image frame. We compute 32, 32 and 16 SFA components for nodes in the lower, middle and top layers, respectively. All connections are topologically organized such that each SFA node receives inputs from neighboring nodes in the preceding layer.

Each individual SFA node implements four subsequent operations, as shown in Figure 3.1 First, a linear SFA is performed which reduces the effective dimension of the node input to 32; then the linear SFA output is quadratically expanded (original data with all quadratic combinations) to introduce non-linearities. White noise is then added to break unwanted redundancies nonlinear expansion might introduce. Finally, a second

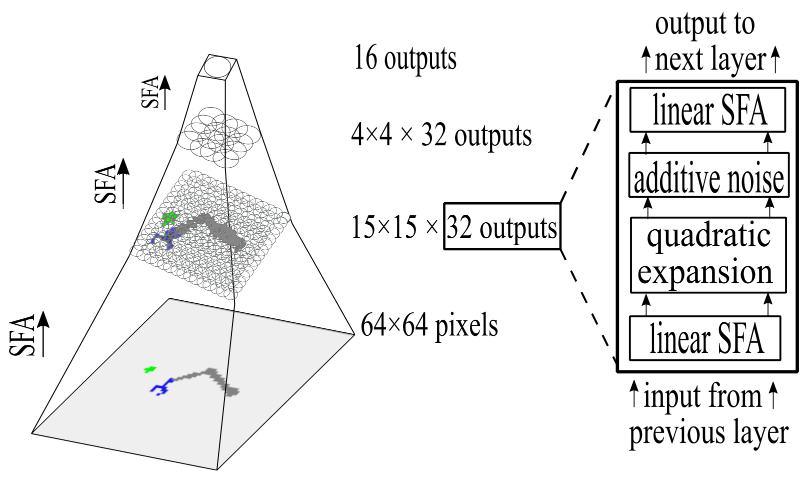


Figure 3.1: Hierarchical SFA network. The network consists of 3 layers of SFA nodes (left). Each node implements 4 sequential operations (right) on the data sequence inside its receptive field from the previous layer and sends its output to the next layer.

linear SFA is performed on the noise-added output. For training the network, we used 50K time points that correspond to a temporal sequence of input images (collected over 3.2 hours on the simulator) generated by a random walk described in Section 3.2.3. The training is done sequentially from bottom to top, each layer at a time using the same training sequence. Once trained, the network is used to generate efficient state representations for our actor-critic learner, as discussed in the following section.

3.2.2 SFA-based ICAC

In Section 2.3, we proposed ICAC, an actor-critic RL algorithm that learns an optimal control policy by learning an ensemble of local predictive models of environment dynamics online and generates an intrinsic reward based on the learning progress of each model. ICAC has only been applied to low-dimensional inputs in non-visual control tasks. Here, we extend and evaluate ICAC on learning vision-based motor control tasks using state representations learned unsupervised with the hierarchical SFA network (Figure 3.1). ICAC, as shown in Figure 3.2, has two components: a growing self-organizing network and a control module. The first incrementally partitions the sensory space into local regions with local predictive models using the Instantaneous Topological Map (ITM) (Jockusch and Ritter, 1999). The network generates an intrinsic reward based on the learning progress of the local model corresponding to the current sensory region. The control module guides the action selection using the combined extrinsic and intrinsic reward as follows: We compute the change between two consecutive average prediction errors of the local model associated with the best-matching node *n* of the ITM

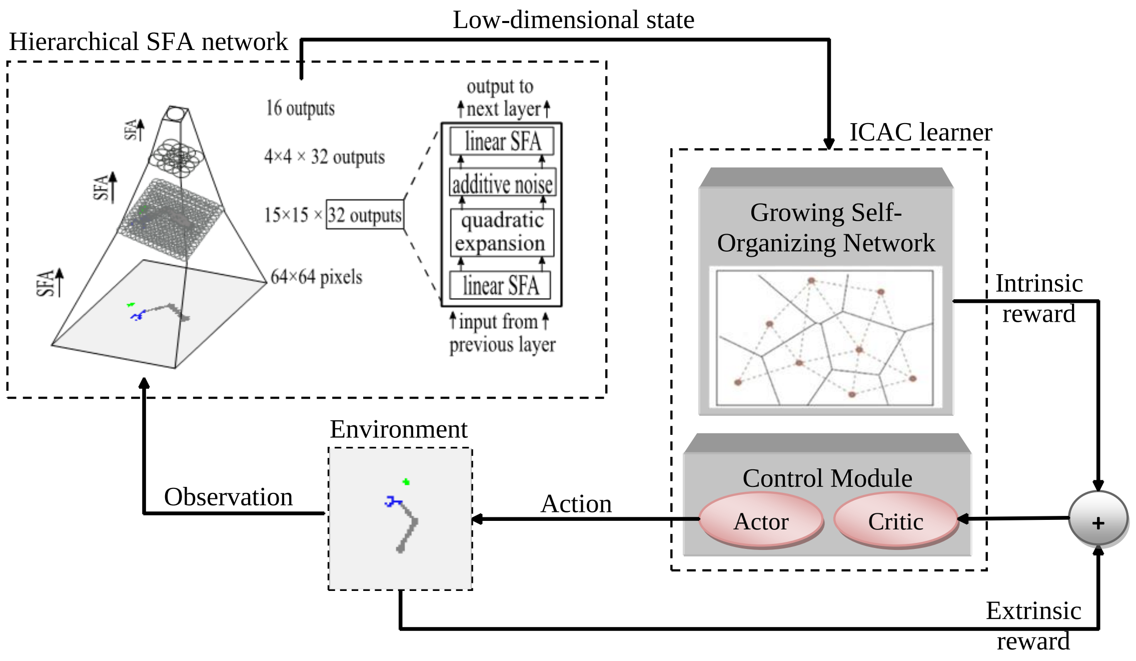


Figure 3.2: SFA-based ICAC system. The ICAC learner takes an action chosen by the actor at the current state and the environment returns an observation. The SFA network takes the new observation and outputs a low-dimensional state encoding which the growing self-organizing network uses to adapt its topology. The learning progress of the local model corresponding to the best-matching node w.r.t. the current state is then computed and used to derive the intrinsic reward that is combined with the extrinsic reward, if any, and fed to the critic to update its estimate of the expected value of the current state. Finally, the actor is updated towards the current action if it is found to lead to higher than expected value.

w.r.t. the current state (the SFA output). This change represents the learning progress the agent has made or expects to make and is combined with the perception error, which is the Euclidean distance between the state encoding and the weight vector of *n*, to give the intrinsic reward. This reward encourages the agent to try actions that maximize its learning progress and lead to perceptually novel states. The Continuous-Actor-Critic Learning Automaton (CACLA) (Van Hasselt, 2012) is used as the base actor-critic algorithm for learning an optimal control policy based on the combined intrinsic and extrinsic reward. The actor is updated towards the current exploratory action, if it leads to a value higher than the critic's estimate. Both the critic and actor are represented by feed-forward neural networks and updated online from the transitions the robot samples while interacting with the environment.

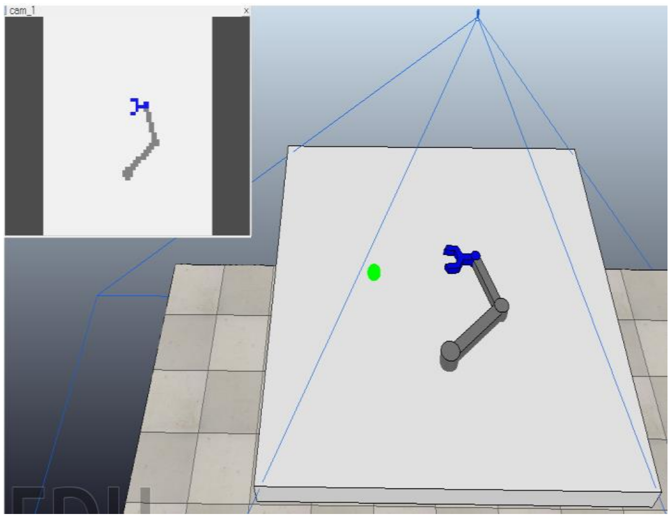
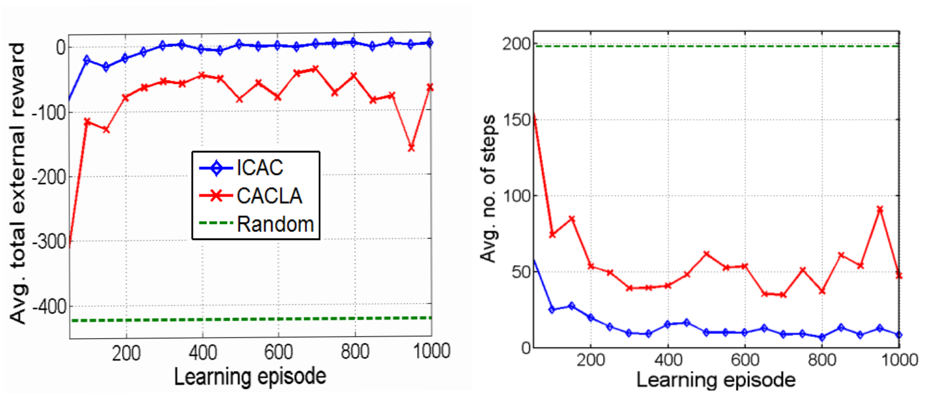


Figure 3.3: The 3D simulation environment with a 3-DoF robotic arm. The vision sensor output is shown in the upper left corner (the goal is not rendered in the input to the SFA).

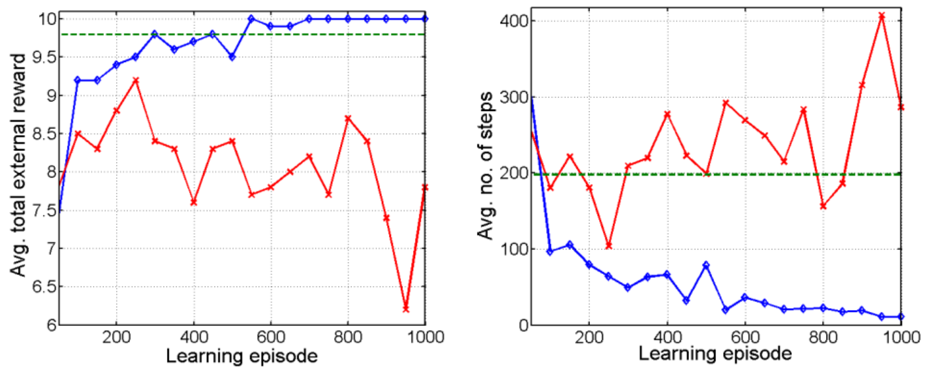
3.2.3 Experiments

We evaluate SFA-based ICAC on a vision-based target reaching task with 3-DoF robotic arm shown in Figure 3.3 in the V-REP robot simulator (Rohmer et al., 2013). The arm's joints can move within $[\frac{-\pi}{2}, \frac{\pi}{2}]^3$ representing the joint values. The robot gets a reward of 10.0 on reaching the goal region (green sphere in Figure 3.3) or, otherwise, a negative reward based on the Euclidean distance of the gripper tip to the goal. We also performed experiments on the more challenging sparse reward setup (reward of 10.0 on reaching the goal region or 0.0 elsewhere). To train the SFA network, we performed random walk of 50K steps (3.2 hours on the simulator). In each step the robot takes a random action (max of 1 degree per joint) in the simulator and records an image of the world. The image sequence is then used as a single batch to train the network on three passes corresponding to the three layers of SFA nodes (see Section 3.2.1).

The actor and critic in ICAC and CACLA are represented by 2-layer, fully connected MLPs of 20 tanh hidden units. The output units are linear: three in the actor and one in the critic. The input to the actor and critic is the 16-dimensional slow features (the output of the SFA network) together with the goal's Cartesian coordinates. The learning rate is set to 0.01. We store the most recent 1 million state transitions in an experience replay buffer and perform mini-batch stochastic gradient descent with batch size 1K for updating the critic during online training. The reward discount factor is set to 0.99. All actions (joint value changes) are drawn from a Gaussian distribution with a mean at the actor's present output and standard deviation of 0.3 and are clamped to a max of $|10.0|$ degrees



(a) Dense reward environment



(b) Sparse reward environment

Figure 3.4: Learning curves of ICAC and the baselines on vision-based robotic reaching with 3-DoF arm in two reward settings: (a) dense reward and (b) reward-sparse. The average over 50 episodes is shown for readability.

Table 3.1: Mean extrinsic reward (upper half) and mean number of steps to the target (lower half) over the entire learning period (learning speed) and over the last 200 learning episodes (final performance).

	Learning speed			Final performance		
	Random	CACLA	ICAC	Random	CACLA	ICAC
Dense reward	-423.58	-85.93	**-7.81**	-434.70	-97.55	**0.87**
Sparse reward	**9.80**	8.07	9.64	9.95	7.45	**10.00**
Dense reward	198.15	58.17	**15.20**	201.08	63.26	**10.63**
Sparse reward	198.15	236.58	**57.65**	201.08	250.50	**30.82**

per joint. The local world models are 2-layer MLPs trained to predict future states (SFA-output) from previous state-action pairs. The above values were determined empirically. In each learning episode, the robot is given a maximum of 1K steps to reach the current goal after which a new episode begins with a random goal position.

We run ICAC and two baselines of CACLA and uniform random policy for 1K episodes and average the results over 10 random seeds. As shown in Figure 3.4, ICAC reached much better policies in a much smaller number of episodes in both reward settings. While CACLA was able to converge to a relatively good policy in the dense reward environment, it failed in the sparse reward environment where ICAC was converging to a near optimal policy after 600 episodes of learning. In Table 3.1, we give a more concise comparison. Again, the table shows that ICAC outperformed the baselines and that its performance was significantly higher in the difficult sparse reward setting, reaching a success rate of 100% over the last 200 episodes.

3.3 ICAC with Deep Convolutional Autoencoder

In this section, we present our actor-critic approach for learning vision-based robot control using jointly optimized state representations and an intrinsic motivation based on the spatially and temporally learning progress for guiding exploration. We first describe our proposed deep learning architecture for jointly training low-dimensional state representations. We then present our Deep ICAC algorithm that uses state representations learned with our proposed architecture and the learning progress-based intrinsic motivation. Finally, we show the results of our robotic experiments in simulation and in the real world.

3.3.1 Jointly Optimized Representations with Actor-Critic-Autoencoder Architecture

In high-dimensional state spaces, the agent requires representations capable of recognizing states that lead to high rewards in order to learn a good value function that makes learning the target policy easier. To support this, we propose a multi-output deep architecture composed of a convolutional autoencoder, critic and actor networks. The autoencoder consists of a convolutional encoder f and decoder g parametrized by ω and $\widetilde{\omega}$ respectively. The critic V is parametrized by $\{\omega, \theta^V\}$ and outputs an estimate of the expected value of a given state s_t. The actor μ is parametrized by θ^μ and outputs an estimate of the optimal action at a given state s_t. The bottleneck layer of the autoencoder is fully connected to the critic and actor networks. The low-dimensional hidden representation at the bottleneck layer $\phi_{s_t} = f(s_t|\omega)$ is jointly trained to reconstruct the original input state and predict its value. The architecture is show in Figure 3.5.

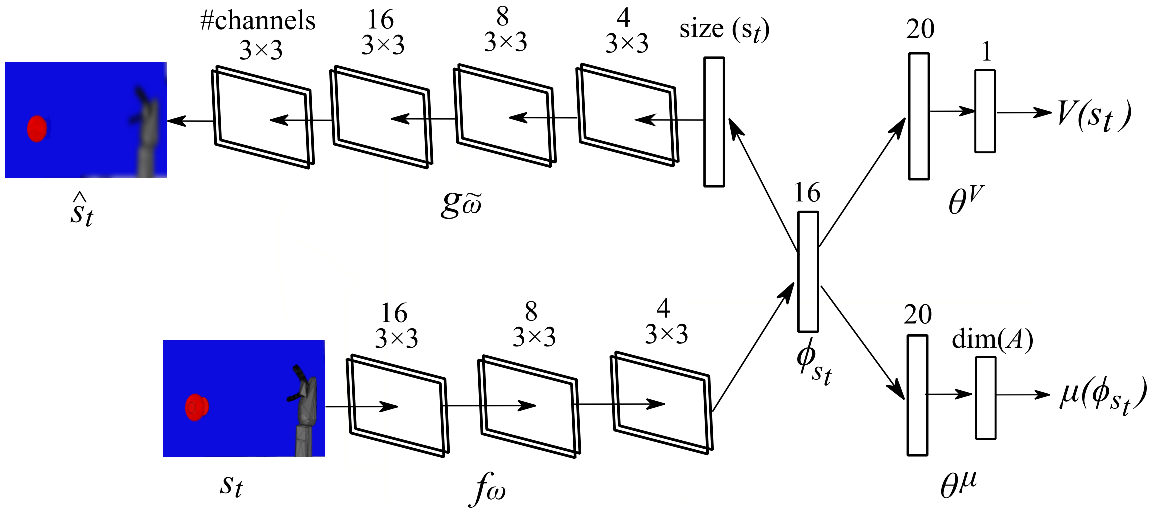


Figure 3.5: The proposed learning architecture: The architecture consists of (1) a convolutional encoder branch f_ω that takes in a raw image s_t and extracts a feature vector ϕ_{s_t}, (2) a convolutional decoder branch $g_{\tilde{\omega}}$ that produces a reconstruction $\hat{s}_t$ of the input, (3) a critic branch V with parameters θ^V that estimates the expected value using the features ϕ_{s_t}, and (4) an actor branch μ with parameters θ^μ that outputs a current estimation of the optimal action $\mu(\phi_{s_t})$ with a dimensionality of dim(A), where A is the action space.

The autoencoder learns the parameters $\{\omega, \tilde{\omega}\}$ that minimize the reconstruction loss between the original input s_t and the reconstructed output $\hat{s}_t = g\left(\phi_{s_t} | \, \tilde{\omega}\right)$:

$$\mathcal{L}_c(\omega, \tilde{\omega}) = \left\| g\left(\phi_{s_t} | \tilde{\omega}\right) - s_t \right\|_2^2. \tag{3.7}$$

The critic learns the parameters $\{\omega, \theta^V\}$ that minimize the value prediction loss between the target value $y_t = r_t + \gamma V'\left(\phi_{s_{t+1}} | \omega', \theta^{V'}\right)$ and the predicted value $V\left(\phi_{s_t} | \omega, \theta^V\right)$:

$$\mathcal{L}_v(\omega, \theta^V) = \left(y_t - V\left(\phi_{s_t} | \omega, \theta^V\right)\right)^2, \tag{3.8}$$

where V' is the target critic network parametrized by $\{\omega', \theta^{V'}\}$. The target network parameters are slowly updated towards their corresponding parameters $\{\omega, \theta^V\}$ to provide more stationary targets, as discussed in Section 1.2.1. The actor learns the parameters θ^μ that move its output closer to an exploratory action a_t found to lead to higher than expected value (i.e. positive TD error). This is done by minimizing the following loss:

$$\mathcal{L}_a(\theta^\mu) = \left(a_t - \mu\left(\phi_{s_t} | \theta^\mu\right)\right)^2. \tag{3.9}$$

The proposed deep architecture is trained online by sampling a minibatch of transitions (s, a, r, s') from the experience replay buffer and performing minibatch stochastic gradient descent to adjust the parameters $\{\omega, \tilde{\omega}, \theta^V, \theta^{AC}\}$ to minimize the joint loss:

$$\mathcal{L}(\omega, \widetilde{\omega}, \theta^V, \theta^\mu) = \mathcal{L}_c(\omega, \widetilde{\omega}) + \mathcal{L}_v(\omega, \theta^V) + \mathcal{L}_a(\theta^\mu). \tag{3.10}$$

This is done by first updating the parameters $\{\omega, \widetilde{\omega}, \theta^V\}$ to minimize the joint unsupervised reconstruction and supervised value prediction loss $\mathcal{L}_c(\omega, \widetilde{\omega}) + \mathcal{L}_v(\omega, \theta^V)$. After that, we fix $\{\omega, \widetilde{\omega}, \theta^V\}$ and update the actor parameters θ^μ to minimize $\mathcal{L}_a(\theta^\mu)$ of the transitions for which the TD error is positive. This iteration ensures that no gradients are backpropagated from the actor branch to affect ω. The state representation ϕ_{s_t} in the proposed architecture is learned to be state discriminator and value predictor by sharing the parameters ω between the encoder and the critic. Using this low-dimensional, task-relevant state representation as a direct input to the actor network allows for more efficient learning of the target policy.

3.3.2 Deep ICAC

Here, we describe how we extend our ICAC algorithm, presented in Section 2.3, to learning control policies from raw pixels using state representations learned with our proposed jointly trained deep neural architecture.

Deep Feature-Space Self-Organization

We incrementally partition the space of jointly learned deep feature representations ϕ_{s_t} (Figure 3.5) into local regions using a growing self-organizing network. We use here the ITM (Jockusch and Ritter, 1999) network, which starts with two connected nodes n_1 and n_2, with weight vectors w_1 and w_2 respectively, and performs the following adaptation steps each time a new stimulus ϕ_{s_t} is observed:

1. Matching: Find the nearest node n and the second-nearest node n' to ϕ_{s_t}: $n \leftarrow \arg\min_i \left\| \phi_{s_t} - w_i \right\|_2^2$, $n' \leftarrow \arg\min_{j, j \neq n} \left\| \phi_{s_t} - w_j \right\|_2^2$.
2. Edge adaptation: If n and n' are not connected, add an edge between them. For all the nodes m connected to n, if n' lies inside the Thales sphere through m and n (the sphere with diameter $w_m w_n$), $i.e. (w_n - w_{n'}) \cdot (w_m - w_{n'}) < 0$, remove the edge between m and n, and if m has no remaining edges, remove m.
3. Node adaptation: If ϕ_{s_t} lies outside the Thales sphere through n and n', $i.e. \left(w_n - \phi_{s_t}\right) \cdot \left(w_{n'} - \phi_{s_t}\right) > 0$, and if $\left\| \phi_{s_t} - w_n \right\|_2^2 > e_{max}$, where e_{max} is the desired mapping resolution, create a new node v with $w_v = \phi_{s_t}$ and an edge with n.

Learning Progress-based Intrinsic Motivation

Each local region n (node in the network) is assigned a distinct predictive model of the world $\mathcal{M}_n$, with parameters $\theta^{\mathcal{M}_n}$. The local models are trained in the space of deep state representations to predict the next representation given the current representation and action. A moving window average of the model prediction error is computed separately for each region *n*:

$$\langle e_{t,n}^{prd} \rangle = \frac{1}{\sigma} \sum_{i=1}^{\sigma} e_i^{prd} \,\Big|_{e_i^{prd} = \left\| \mathcal{M}_n\left(\phi_{s_i}, a_i | \theta^{\mathcal{M}_n}\right) - \phi_{s_{i+1}} \right\|_2^2}, \tag{3.11}$$

where σ specifies the length of the window of recent predictions in *n*. The improvement in model predictions, the change in $\langle e_{t,n}^{prd} \rangle$ over time, is then estimated by computing the learning progress (LP) locally in each region using a time window $\mathcal{W}$:

$$LP_{t,n} = \left| \langle e_{t-\mathcal{W},n}^{prd} \rangle - \langle e_{t,n}^{prd} \rangle \right|. \tag{3.12}$$

After taking an action a_t at state s_t and observing the next state s_{t+1}, we update the local learning progress that corresponds to the region *n* to which the state representation ϕ_{s_t} belongs using Equations 3.11 and 3.12. The updated learning progress $LP_{t,n}$ is then combined with the perception error e_t^{per}, which is the distance between the state representation $\phi_{s_{t+1}}$ and the weight vector of the best-matching node *m* of the ITM w.r.t. to $\phi_{s_{t+1}}$, to give the intrinsic reward signal:

$$r_t^{int} = LP_{t,n} + e_t^{per}. \tag{3.13}$$

The intrinsic reward directs exploration towards perceptual novel regions of the space of deep state representations and where maximum learning progress is expected. To use the derived intrinsic reward in our actor-critic system, we gradually anneal it to account for the fact that with more interactions the agent becomes less uncertain about its world dynamics. We combine it with the extrinsic reward as follows:

$$r_t = r_t^{ext} + \frac{r_t^{int}}{1 + D \cdot t}, \tag{3.14}$$

where $D > 0$ is a decay constant. At each time step *t*, an action $\mathrm{a_t}$ is drawn from a conditionally Gaussian policy $\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(a_t - \mu(s_t|\theta^{\mu})\right)^2/2\sigma^2}$ centered at the actor's output $\mu(s_t|\theta^{\mu})$ with a standard deviation σ. Figure 3.6 shows the overall learning system, demonstrating the interaction among the different components of our approach. The learning algorithm is detailed in Algorithm 2.

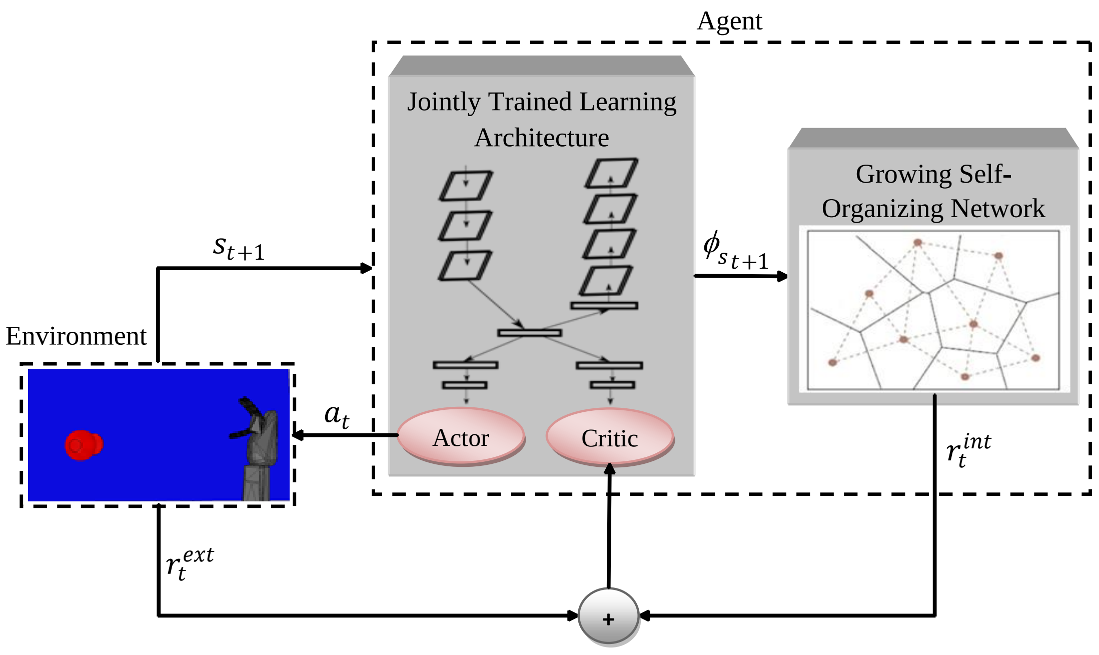


Figure 3.6: Deep ICAC learning system: The agent takes an action a_t sampled from a Gaussian policy centered at the actor's output, and the environment returns a new state s_{t+1} and a reward r_t^{ext}. The convolutional encoder of the jointly trained learning architecture then computes a feature representation $\phi_{s_{t+1}}$ which the growing self-organizing network ITM uses to adapt its topology. The learning progress of the local world model corresponding to the ITM's best-matching node w.r.t. ϕ_{s_t} is then computed and used to derive an intrinsic reward r_t^{int}. The intrinsic reward is combined with the extrinsic reward r_t^{ext} and fed to the critic to update its estimate of the utility of a_t. Finally, the actor is updated towards a_t if it is found to lead to higher than the critic's estimated value.

3.3.3 Experiments

We evaluate our approach on robotic learning-to-reach and learning-to-grasp tasks. In all the experiments, we compare the proposed Deep ICAC to Deep CACLA (CACLA with our proposed deep architecture) and the state-of-the-art DDPG (see Appendix B). We consider two environmental conditions for each task: dense and sparse reward settings.

Parameters

We employ a convolutional autoencoder that includes 7 zero-padded convolutional layers with ReLU activations, 2 dense layers with ReLU activations, and no pooling layers, as shown in the encoder and decoder branches of Figure 3.5. The figure also shows the number and size of the filters used in each layer. All convolutional layers have the same filter size (3×3) applied with stride 1 to maintain the size of the input image. The critic network consists of the encoder layers followed by a dense layer with 20 ReLU neurons

Algorithm 2 Deep ICAC

```
 1: Initialize the parameters $\{\omega, \widetilde{\omega}, \theta^V, \theta^\mu, \omega', \theta^{V'}, \tau\}$
 2: Initialize the ITM network
 3: Initialize replay buffer $R$
 3: for $episode = 1$ to $E$ do
 4:   Sample initial state $s_1$
 5:   for $t = 1$ to $T$ do
 6:     Sample action $a_t \sim \pi$: $\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(a_t - \mu(s_t|\theta^\mu))^2/2\sigma^2}$
 7:     Execute $a_t$ and observe $r_t^{ext}$ and $s_{t+1}$
 8:     Update the ITM and the model in the deep feature-space region covering $\phi_{s_t}$
 9:     Compute the intrinsic reward $r_t^{int}$ using Equation 3.13
10:     Compute the total reward $r_t$ using Equation 3.14
11:     Store the transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
13:     Update $\{\omega, \widetilde{\omega}, \theta^V\}$ on minibatch from $R$ to minimize $\mathcal{L}_c(\omega, \widetilde{\omega}) + \mathcal{L}_v(\omega, \theta^V)$
14:     Update $\theta^\mu$ on minibatch from $R$, following Equation 3.9
15:     Update target network parameters $\theta^{V'} \leftarrow \tau\,\theta^V + (1-\tau)\theta^{V'}, \omega' \leftarrow \tau\omega + (1-\tau)\omega'$
16:   end for
17: end for
```

and a dense output layer of a single linear neuron. The fourth layer of the encoder is a dense layer with 16 neurons whose output is used as a low-dimensional feature vector ϕ fed as input to the actor network. The actor network is a 2-layer fully connected MLP of 20 tanh hidden neurons. The output neurons have tanh activations (to bound actions) and represent an action vector whose dimension depends on the task. We train the networks with proportional Prioritized Experience Replay (PER) (Schaul et al., 2016) using the Adam optimizer (Kingma and Ba, 2014) and a learning rate of 1e-3 for both the autoencoder and critic and 1e-4 for the actor (see Appendix B). We use a replay buffer of size 100K and a minibatch size of 64 sampled using PER. The PER hyperparameters α and β_0 are set to 0.6 and 0.4 respectively. The target value network's update factor τ is set to 1e-3. The reward discount γ is 0.99. We set the intrinsic reward decay constant D to 0.1. The intrinsic reward is normalized so that it remains in the interval $[0, 1]$. The ITM model has the threshold $e_{\max}$ as its only hyperparameter, which we set to 6. Five nodes, i.e. predictive models, are generated on average. All predictive models used are 2-layer fully connected MLPs of 20 tanh hidden and 16 linear output neurons trained online with Adam optimizer. Exploratory actions are Gaussian distributed with a standard deviation of 20 degrees and a mean at the current actor's output.

Table 3.2: Comparison between the number of learning parameters of the different deep architectures used in the experiments.

	DDPG	Deep CACLA/ICAC
Actor network	36,077,399	403
Critic network	36,077,585	935,716
Total	72,154,984	936,119

The above values were determined empirically based on preliminary experiments and the following findings were obtained. Different numbers for the dense layer neurons of the actor and critic networks made no significant difference to the results. For the bottleneck layer of the autoencoder, we tested the performance for 8, 16, 32, and 64 neurons. Reducing from 16, as finally used, to 8, the average reward decreased to below 2.5. Increasing from 16 to 32 and 64 did not significantly change the average reward. Different learning rates were evaluated and found to slightly affect the learning performance. However, learning rates below 1e-3 for training the autoencoder and critic caused slow learning convergence. Minibatch sizes larger than 64 did not lead to considerable performance improvement. The value of γ did not correlate with the performance. Our own DDPG implementation for learning from pixels uses the same neural architecture described in (Lillicrap et al., 2016) and the best-performing hyperparameters we empirically found, in addition to training with proportional PER. A comparison between the number of learning parameters used in the proposed neural architecture and that of DDPG is presented in Table 3.2.

Parameter Choice Analysis

While the structural and learning parameters of our proposed deep architecture is based on standard deep learning models and so their choice can be directly understood, some other parameters are less straightforward. Here, we particularly explain the role and choice of the PER, ITM and intrinsic reward decay parameters as follows:

- In PER, transitions are sampled from a replay buffer with probability proportional to their priorities $P(i) = p_i^\alpha / \sum_k p_k^\alpha$ where p_i is the priority of transition i represented by the absolute value of its TD error and the exponent α determines the amount of prioritization used, with $\alpha = 0$ corresponding to the uniform random sampling. The larger the value of α the stronger is the prioritization. The prioritization introduces a bias by changing the distribution of the transitions used for learning. To compensate for the bias, importance-sampling weights are used $w_i^{PER} = 1/(N \cdot P(i))^\beta$ where N is the buffer size. Full compensation corresponds to $\beta = 1$. These weights are multiplied by the TD error when updating the value function parameters. The bias is less significant prior to convergence, since the policy and state distribution are non-stationary. Therefore, β is usually annealed from some initial value β_0 to reach 1 at the end of learning. We empirically found $\alpha = 0.6$ and $\beta_0 = 0.4$ to yield stable results in all our experiments.

- In ITM, a new node is created when the stimulus is more than a given threshold e_{max} away from the nearest node. This means that e_{max} determines the desired mapping resolution as it controls the growth of the ITM map. The choice of e_{max} can influence the derived intrinsic reward by affecting the number of local predictive models generated. The results of setting e_{max} to 6.0 were on average better than other values we experimented with. Smaller values increased the computation time without significant performance gain.
- In the combined reward signal derived in Equation 3.14, the parameter D controls the decay rate of the weight of the intrinsic reward component r_t^{int}. Reasonably small values for D keep the agent more intrinsically motivated during the early stages of learning while allowing it to become gradually less intrinsically motivated as it learns more about the world dynamics and its action values. We found $D = 0.1$ as the best performing value in our experiments.

Vision-based Learning-to-Reach

We evaluate our approach on the learning-to-reach task using the V-REP robot simulator (Rohmer et al., 2013). The 3-D robotic environment used in the conducted experiments is shown in Figure 3.7. The environment consists of a 3-DoF robot arm with a gripper attached and a red cylindrical target object. A vision sensor is used and positioned vertically above the scene to capture real-time 84×84 pixel RGB images of the states of the environment. Each joint of the robot can move in the angular range of $[-\pi/2, \pi/2]$. A reaching attempt is considered successful when the gripper center is within a predetermined radius from the center of the target object (the resulting target zone area is equivalent to 9% of the total reachable area). The reward function used is as follows:

$$r_t^{ext} = \begin{cases} +10 & if\ successful, \\ -\|c^t - c^g\| & otherwise, \end{cases}$$

where $\|c^t - c^g\|$ is the Euclidian distance between the center of the target object c^t and

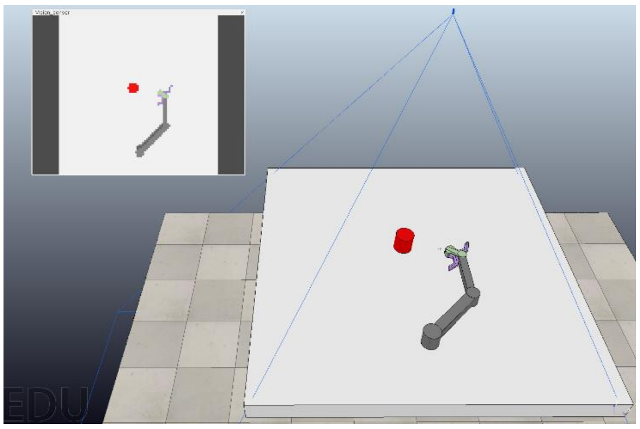
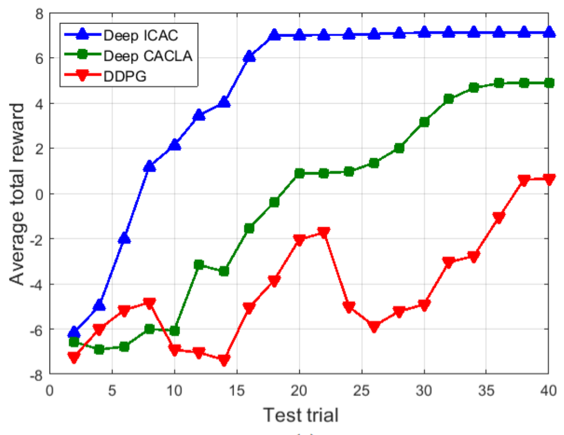


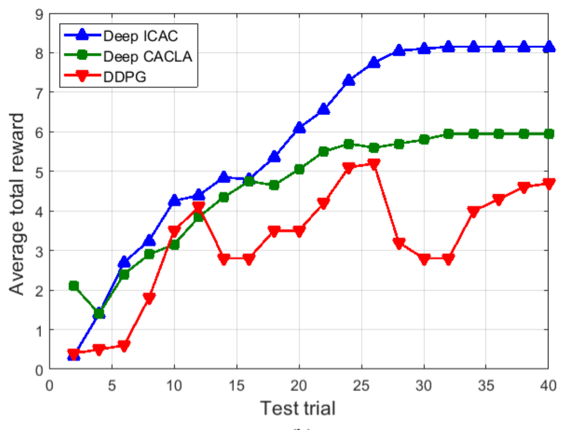
Figure 3.7: The V-REP simulation environment for the task of learning-to-reach including the 3-DoF arm with a gripper attached and a red cylindrical target. The vision sensor output is shown in the upper left corner.

the center of the gripper c^g. In the experiment with sparse rewards, a reward of 0 is given for unsuccessful actions. We ran Deep ICAC, Deep CACLA, and DDPG on dense reward and sparse reward environments for 10K episodes with a maximum of 10 steps per episode, with the position of the target object varying randomly every episode. Training was done by sampling from the replay buffer with PER and performing a mini-batch stochastic gradient descent optimization with Adam. For evaluating the learned policy, training was paused after every 250 episodes and a test trial was performed that includes running the policy without any updates for 20 episodes each with a different target position not included in the training. The average total (extrinsic) reward over the 20 test episodes was then reported for every test trial. We ran all the experiments on a single Nvidia GTX 1050 GPU with an average runtime of five hours per run for each of the algorithms considered.

Figure 3.8 shows the results of applying the algorithms to the environment in the dense reward and the sparse reward settings. The results shown are averages over 20 seeds. The performance of the learned policy was almost identical among the three algorithms during the first five test trials (1K learning episodes) in the dense reward setting and the first ten trials (2.5K learning episodes) in the challenging sparse reward setting. However, only the policies learned with Deep ICAC and Deep CACLA continued to improve steadily with Deep ICAC converging faster to an average return of 7.1 in the dense reward setting and over 8.0 (i.e. success rate of over 80%) just below the optimal policy (return of 10) in the sparse reward setting. Despite its good performance in the dense reward setting, Deep CACLA suffered from a premature convergence to a locally optimal



(a)



(b)

Figure 3.8: Performance curves of Deep ICAC, Deep CACLA, and DDPG on the robotic learning-to-reach task in different reward settings: (a) dense reward and (b) sparse reward.

Table 3.3: Learning statistics in the learning-to-reach experiment with dense rewards (upper half) and sparse rewards (lower half).

	DDPG	Deep CACLA	Deep ICAC
Learning speed	4.52	**6.52**	6.11
Final performance	4.08	7.01	**7.51**
Learning speed	5.34	7.25	**8.14**
Final performance	5.1	6.8	**9.0**

policy in the sparse reward setting. DDPG, on the other hand, showed poor stability unable to reach a good policy by the end of the training process in both reward settings. We also report learning statistics in terms of the average reward per episode over the entire training process (learning speed) and over the last 100 episodes of training (final performance) in Table 3.3. The data shown are averages over 20 runs.

Vision-based Learning-to-Grasp

In the second experiment, we consider robotic grasping as a learning task. Unlike reaching, grasping requires more precise motor actions, handling of external collisions with the object to grasp, and finding correct finger placement. The robotic environment consists of our Neuro-Inspired COmpanion (NICO) humanoid (Kerzel et al., 2017) facing a table on top of which a target object is placed. Figure 3.9 shows the V-REP simulation scene of the experiment. To avoid self-collisions while allowing for a larger task space for grasp learning, we consider a motor policy involving the right shoulder joint and the right-hand joints, as shown in Figure 3.10(a). NICO's right arm has 6 DoF of which we control one in the shoulder. The shoulder joint can move in the angular range [-100,100]

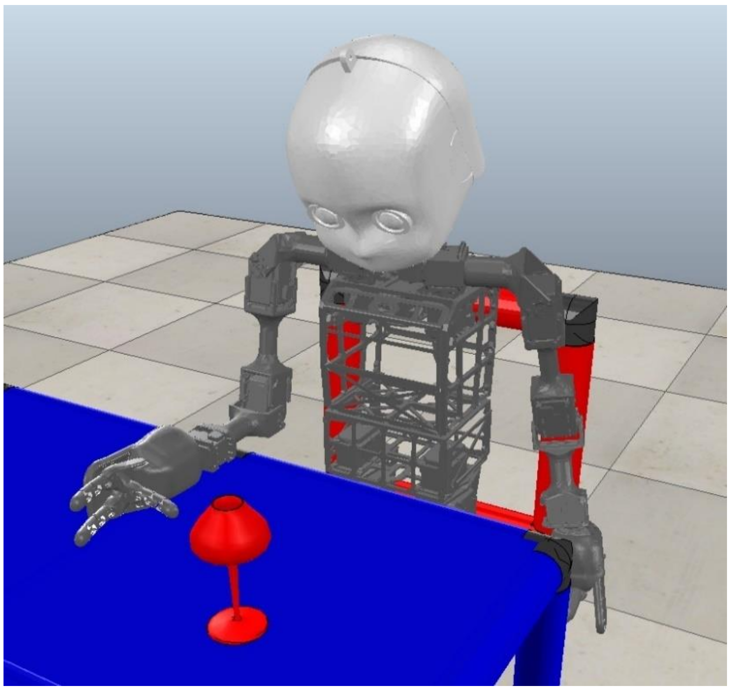


Figure 3.9: The V-REP simulation environment used in the learning-to-grasp experiment including the NICO robot sitting in front of a table on top of which a target object is placed. NICO learns to grasp the object with its right multi-fingered hand.

(in degrees). NICO's hand is 11-DoF multi-fingered with two index fingers and a thumb each of which can move in the angular range [-160, 160] (in degrees). The robot learns to control 2 DoFs: 1 DoF (shoulder joint) and 1 DoF (hand open/close). The only input to the learning algorithm is the raw data of 32×64 pixel RGB image, which is used as the state of the environment, obtained from the vision sensor output shown in Figure 3.10(b). We use the following reward function:

$$r_t^{ext} = \begin{cases} +10 & if\ successful, \\ -10 & if\ object\ is\ toppled, \\ -\|c^t - c^h\| & otherwise \end{cases}$$

where c^t is the center of the target object and c^h is the center of the robot hand. We determine successful grasps by moving the shoulder joint 20 degrees in the opposite direction of the recently applied joint value and measuring the Euclidean distance $\|c^t - c^h\|$ afterwards. If the distance remains below a grasp threshold of 0.04 m, the grasp is deemed successful. Otherwise, the hand is opened, the shoulder joint moves back to its previous value, and the robot continues the learning episode. In the sparse reward setting, we use the following sparse reward function:

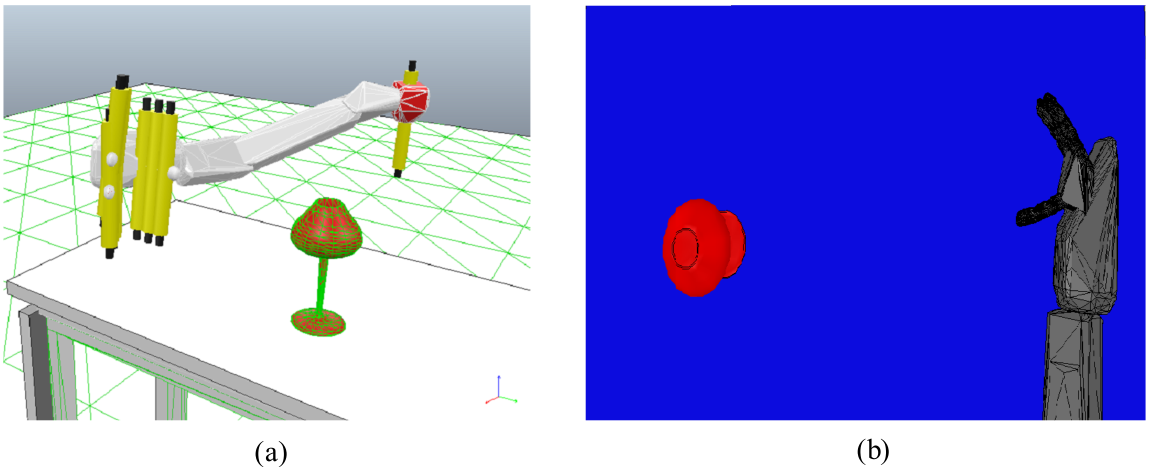
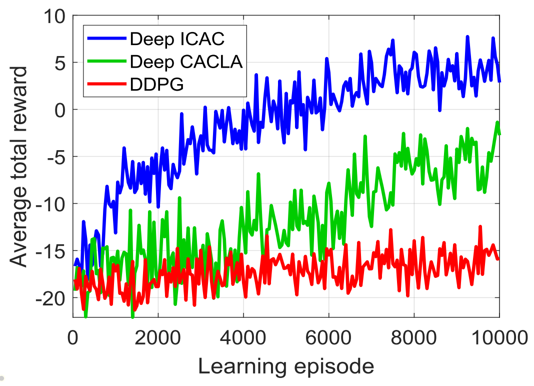


Figure 3.10: (a) The raw motor output and (b) raw sensory input considered in the learning-to-grasp experiment. Yellow cylinders in (a) refer to the axes of rotations of the joints controlled during grasp learning.

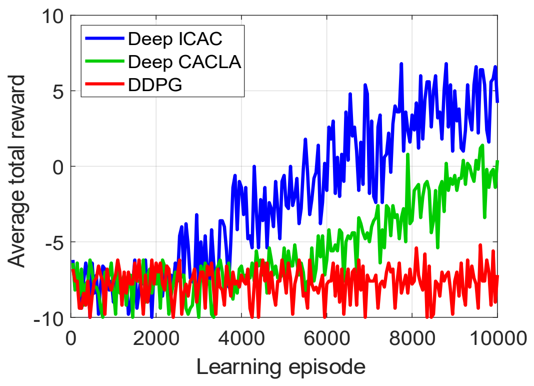
$$r_t^{ext} = \begin{cases} +10 & \text{if successful,} \\ -10 & \text{if object is toppled,} \\ 0 & \text{otherwise.} \end{cases}$$

We run the algorithms for 10K episodes with a maximum of 50 actions per episode and with the target object randomly placed in a graspable position at the start of each episode. The episode terminates when the object is successfully grasped, the object is toppled, or a maximum number of 20 action steps is reached.

The learning-to-grasp experiments were run on a single Nvidia GTX 1050 GPU with an average runtime of ~25 hours per run for all the algorithms in the dense reward setting. In the sparse reward setting, the average runtime was 27.2, 33.8, and 35.5 hours for Deep ICAC, Deep CACLA, and DDPG respectively. Figure 3.11 shows the average total extrinsic reward per learning episode over five seeds. Gradual performance improvement was observed for all the algorithms in the environment with dense reward setting, as shown in Figure 3.11(a). Starting at around an average total reward of -17, Deep CACLA and Deep ICAC reached a policy with an average return of 0 and 5 respectively. The DDPG progress, on the other hand, was very slow, moving from -18 to -15 by the end of the learning process. In the sparse reward environment, the algorithms were unable to make notable progress for 3K episodes after which the learned policy of only Deep ICAC and Deep CACLA improved while DDPG's remained the same.



(a)



(b)

Figure 3.11: Learning curves of Deep ICAC, Deep CACLA, and DDPG on the robotic learning-to-grasp task in different reward settings: (a) dense reward and (b) sparse reward. The average over 50 episodes is shown for readability.

Vision-based Learning-to-Grasp on the Physical NICO

Deep RL is well suited for research on physical, developmental robots (Cangelosi and Schlesinger, 2015). Enabling robots to learn increasingly complex sensorimotor abilities through interaction with the real environment would move the state of the art in robotics from laborious programming tasks that can only be realized by highly specialized experts into the realm of intuitive, human-like teaching scenarios, or even robots, that can carry out repetitive learning tasks autonomously. To realize this, several obstacles have to be overcome: Deep RL requires a large number of samples. Successful application of deep RL has been achieved for games (Mnih et al., 2015) and purely virtual environments (Lillicrap et al., 2016). In virtual environments, a large number of samples can be collected within a short time, without the danger of damaging the learner or the environment and without human assistance or supervision. A simulation can be reset to its initial state, whenever an unwanted state occurs. Likewise, any required change to the environment or assistance can be automated. An example could be lifting up a toppled object and putting it back into the robot's reach. For a developing child, these chores are usually realized by its caretakers: in a typical parent-child interaction, the child learns under the supervision of adults who provide a safe environment that enables suitable learning steps. Therefore, when moving to a real robot the research question is twofold: The core research question is the evaluation of the Deep ICAC algorithm on a real robotic system. We analyze how real sensor and motor noise affect the learning outcome. The secondary research question is the design of an experimental setup that enables autonomous learning, i.e. learning without constant human assistance.

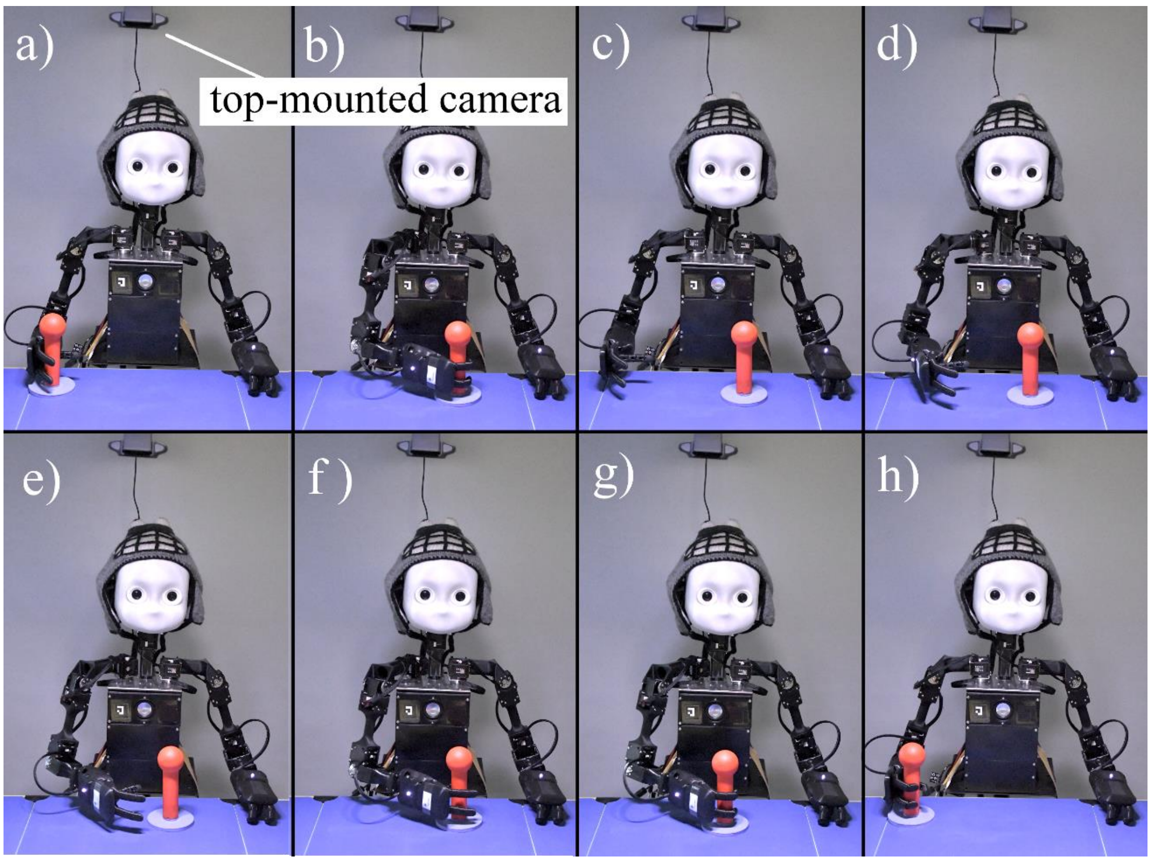


Figure 3.12: NICO experimental setup during learning including a red object for grasp-learning and a top-mounted camera. The experiment starts with NICO's hand at its start position (a). Using its shoulder joint, NICO grasps and moves the object to a random target position which is then recorded (b). Next, NICO moves back the hand to the home position (c). Learning starts by taking the image provided by the top-mounted camera as an input and producing an action output from the actor network of the Deep RL algorithm. A sequence of actions is mostly required to reach and grasp the object since the maximum angle change of the joint is limited (d-f). NICO closes its hand when the algorithm recognizes that the object has been reached (g). Once the object is grasped, the hand with the object grasped is moved to the home position and the learning cycle is repeated (h). In case of reaching a maximum of 50 action steps, the shoulder joint position is set to the recorded target position to grasp the object and move it to the home position before repeating the learning cycle.

As a robotic platform, we use NICO (Kerzel et al., 2017), a child-sized humanoid developed by the Knowledge Technology group for research on neurobotic and cognitive learning models and on human-robot interaction. NICO is an open and highly customizable platform. NICO's relevant functionalities for the experimental setup are its 6-DoF arms based on humanoid anatomy and range of motion. NICO has three-fingered hands that are robust and reliable. NICO's arm is articulated with Dynamixel servomotors. As the presented experiments only use the upper body functionality, the experiments are

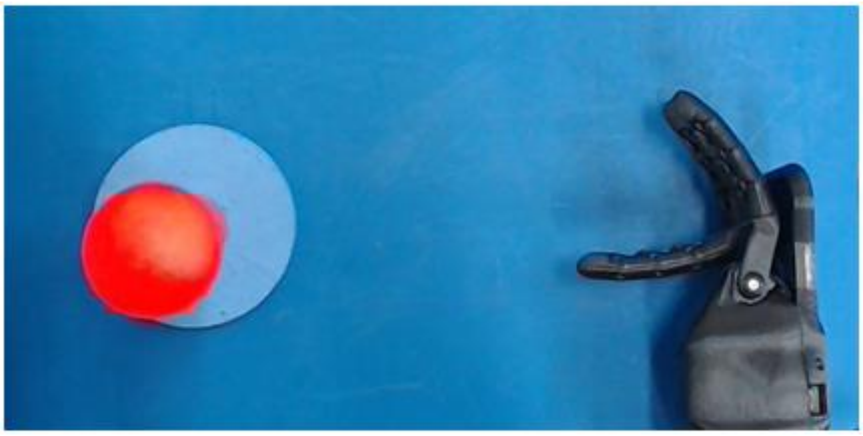


Figure 3.13: The image obtained from the top-mounted camera in the NICO experimental setup.

carried out on the torso version of NICO that is placed in a fixed position as if seated at a table, as shown in Figure 3.12. Though NICO has two integrated cameras in its head and can view its workspace on the table with its articulated head, an external camera was used to mimic the position of the virtual camera from the experiment with simulated NICO to ensure comparability and transferability.

Our physical experimental setup follows the approach by (Kerzel and Wermter, 2017) in which a robot is able to manipulate its environment with simple, non-learned motor actions to provide suitable learning input. To learn to grasp, the robot executes a self-learning cycle depicted in Figure 3.12. Initially, NICO moves the hand to its start position and the grasp-learning object is put into NICO’s hand (a), NICO then grasps the object and moves it to a random position on the table by using only its shoulder joint (b). The joint position is recorded and the object is released, the now empty hand moves back to the home position (c). So far, we have utilized basic robotic motor abilities, now the learning phase begins: The top-mounted camera provides an image to the learning algorithm (see Figure 3.13), and the output of the actor’s neural network is set as the next angular change of the shoulder joint. As a result, NICO moves its hand towards the grasp-learning object (d-f). As the maximum change in the joint angle is limited, mostly several steps are needed until NICO's hand reaches the object. Once the deep RL algorithm recognizes that the hand has reached the grasp-learning object based on the distance between the current and target positions of the shoulder joint, a command to close the hand is generated (g). In the case of a successful grasp, the hand and the held object are moved back to the home position (h) and the learning cycle is repeated. If a maximum number of 50 steps is reached, the hand is opened and the shoulder is moved to the recorded joint position to grasp the object which is then moved to the home position (a). We limit the joints' speed so that we do not have cases where the object is pushed away from NICO's hand or toppled over. In case the object is pushed, it stays inside NICO's open hand, which is then closed on the object, once the motion is finished, and moved to the home position (a). The advantage of this self-learning cycle is the complete independence of external assistance. Basic robotic motion and recording abilities are used to provide learning instances by placing the object at a random position as well as resetting the experiment in the cases where the learned grasp is not successful.

With regard to the learning algorithm, the experiment uses the same parameters as in the virtual environment. The algorithm was trained for 4K episodes with a maximum of 50 actions per episode. A full training of the deep RL approach was conducted without human supervision for over 50 hours, during which about 15K samples were collected. During the self-learning cycle, the grasp-learning object is placed in a random graspable position within the same range of possible positions. 32×64 pixel RGB images from a top-mounted camera are used as visual input. We use the following reward function:

$$r_t^{ext} = \begin{cases} +10 & if\ successful, \\ -10 & if\ object\ is\ pushed, \\ -\|p^t - p^c\| & otherwise, \end{cases}$$

where p^t and p^c are the target and current positions of the shoulder joint respectively. We define a successful grasp as having a distance in the joint space of less than 1.7 degrees. All hyperparameters for the learning algorithms remain unchanged from the experiment presented on the simulated NICO. The results of the learning are presented in Figure 3.14. Compared to the training in simulation, the training on the physical robot shows a very similar learning curve. After 4K learning episodes, the Deep ICAC on the physical NICO is able to reliably grasp objects with 76% grasp accuracy (see Table 3.4).

Time Complexity

One main computational difference between DDPG and our proposed algorithms is the cost of the minibatch gradient descent step during experience replay. While all the algorithms have relatively similar cost for updating the critic network, they have significantly different cost for updating the actor network. DDPG performs a product between the $1{\times}s_a$ vector $\nabla_a Q(s,a|\theta^Q)$ and the $s_a{\times}s_w$ Jacobian matrix $\nabla_{\theta^\mu}\mu(s|\theta^\mu)$ n times, where s_a is the action dimension, s_w is the number of the actor network's weights and n is the

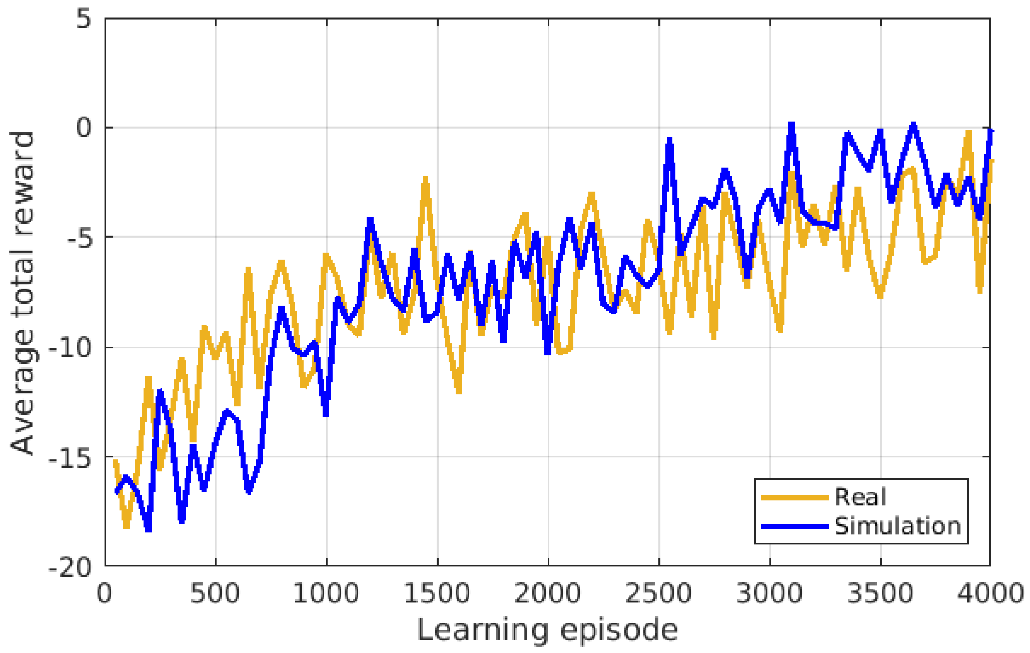


Figure 3.14: Learning curves of Deep ICAC for the vision-based learning-to-grasp task on the simulated and physical NICO robot.

Table 3.4: Test results of running Deep ICAC using the networks trained on the physical NICO.

	No. of Trials	No. of Success	Success Rate
Deep ICAC on physical NICO	25	19	0.760

minibatch size (see Equation 1.8). This gives a complexity of $O(n \cdot s_a \cdot s_w)$. Deep CACLA and Deep ICAC, on the other hand, backpropagate the gradients of the loss in Equation 3.9 computed at the actor's output layer to preceding layers with a complexity of $O(n \sum_{l=1}^{L} s_l \, s_{l-1}) = O(n \cdot s_w)$, where L is the number of layers, s_l is the layer size and the input is the feature vector ϕ_s. Since $s_w \approx 36M$ in DDPG but $s_w \approx 400$ for our actor (see Table 3.2), this means our actor is updated roughly 250K times faster than in DDPG when $s_a = 3$ (even more if $s_a > 3$), benefiting from the small 2-layer architecture trained on the low-dimensional ϕ_s. The overall cost of the minibatch update is linear in the minibatch size and in the number of networks' parameters.

It should be noted that Deep ICAC has an additional cost for updating the ITM network each time a transition is observed. This involves the matching step that scales with the number of nodes and the edge adaptation step that scales with the average number of neighboring nodes. All other operations are independent of the number of nodes. The cost of updating the predictive model of the best-matching node is $O(\sum_{l=1}^{L} s_l \, s_{l-1})$, which is the cost of a backpropagation pass on the 2-layer network and equals 640 per transition in our settings. This added complexity is minimal when the average size of the ITM network is small (5 ITM nodes in our experiments). Consequently, the data efficiency of Deep ICAC does not come at the expense of a greater computational complexity, and this is especially evident in the results of learning grasp policies in real time on the physical robot with Deep ICAC.

Discussion

The obtained results can be summarized as follows: First, Deep CACLA is significantly more stable and learns continuous control policies with high returns faster than DDPG. Second, Deep ICAC is inherently more sample-efficient than both Deep CACLA and DDPG and its superior performance is particularly pronounced in the challenging sparse reward setting. Third, DDPG suffers from poor sample efficiency as well as learning instability, diverging from a good target policy multiple times.

The observed difference in performance between Deep CACLA and DDPG mainly stems from the policy update mechanism and the learned state representation. While DDPG updates the policy by gradient ascent on the currently learned action-value function that is initially not well trained, Deep CACLA updates the policy towards the recent action only when an actual increase in the predicted value is observed. This conservative update results in more stable learning, preventing any significant divergence from the currently best-known policy, as shown in the results. The jointly optimized state representation of Deep CACLA, which is used as an input to the actor, leads to fast learning of better control policies by providing state-discriminative and value-predictive features that are low-dimensional and more accurately recognize states with high expected values. It is clear from the results that both DDPG and, to a lesser extent, Deep CACLA have a slow convergence to a good policy and thus require more training samples. This is largely due to the exploration policy employed which is undirected and leads to more training time spent in parts of the sensory space that are more frequently explored than others. Deep ICAC, on the other hand, provides directed exploration through its learning progress-based intrinsic reward. The intrinsic reward in Deep ICAC prevents spending additional training time in well-explored regions of the sensory space and is more robust to noise and task-irrelevant stochasticity in the environment. This guarantees efficient exploration and fast convergence to near-optimal policies, which is evident in the obtained results.

In the experiments with sparse reward environments, the robot lacks frequent feedback signals important for improving the learned policy, rendering the task more difficult. Therefore, Deep CACLA and DDPG exhibit slower learning performance in such environments than in the environments with dense rewards. Deep CACLA overcomes this difficulty by combining the sparsely available extrinsic reward with its exploration-oriented intrinsic reward, enabling the robot to continue to learn driven by the intrinsic motivation to improve its knowledge about the world.

What distinguishes the learning architecture of Deep CACLA and Deep ICAC is the use of a convolutional autoencoder, rather than a standard Convolutional Neural Network (CNN) commonly used when learning control policies from raw images. A standard CNN requires either standard deep RL with reward-based losses, which is unrealistic given the sparse feedback, or supervised learning with labeled pairs of states and their optimal actions. Conversely, the convolutional autoencoder can be trained unsupervised from the available images with a rich error signal, allowing seamless integration of unsupervised and RL training objectives, as detailed in Section 3.3.1.

Our approach learns action policies purely end-to-end without any prior knowledge or assumptions about the geometry of the robot, its environment, or the appearance of the target object in all the conducted experiments. Also, no knowledge of the kinematics of the robot and the pose of the target object is assumed. Our intrinsic reward module is general enough to be potentially used in a variety of RL methods, including value-based methods and policy gradient methods (deterministic, e.g., DDPG or stochastic, e.g., A3C (Mnih et al., 2016)). Our experiments show that the Deep ICAC algorithm enables the physical robot to successfully learn a visuomotor skill without human assistance during the self-learning phase. The learned skill is limited to two degrees of freedom, but this limitation is in line with the developmental robotics paradigm of learning increasingly complex skills, which is also found in other areas of artificial neural learning (Elman, 1993). Based on the acquired skill, more complex skills can follow as each learned ability adds to the toolbox of abilities that can be used in the next learning setups.

3.4 Conclusion

In this chapter, we proposed two neural architectures for learning efficient and task-relevant state representations from high-dimensional observations. First, we presented the SFA-based ICAC algorithm for learning visuomotor control policies using an intrinsic reward based on the spatially and temporally local learning progress and using state representations learned unsupervised with a hierarchical SFA network. The slow features learned with our SFA network provide a low-dimensional representation that encodes the important invariances in the raw observation. The results show that our SFA-based ICAC can achieve better performance than the actor-critic baseline and converge to a near-optimal control policy in a relatively small number of training episodes in dense and sparse reward environments. We then presented the Deep ICAC algorithm that uses state representations learned with our proposed jointly trained deep neural architecture. The architecture is composed of a convolutional autoencoder, actor and critic networks. The hidden representation at the bottleneck layer of the autoencoder is trained to minimize the unsupervised reconstruction loss of the autoencoder and the supervised value prediction loss of the critic, and therefore, it captures the information necessary to reconstruct the original input and recognize states that lead to high rewards. The actor network takes this jointly optimized low-dimensional representation as a direct input, which enables faster learning of the target policy. The algorithm trains local world models in the space of the learned state representations and uses their predictions in computing the learning progress-based intrinsic reward for directed exploration. We evaluated the Deep ICAC algorithm on learning robot reaching and grasping skills from raw pixels in simulation and in the real world. The results show that Deep ICAC outperforms the state-of-the-art and baseline algorithms in terms of learning speed and final performance in both dense and sparse reward environments. The real-world experiment demonstrates that Deep ICAC can effectively be used for vision-based control learning on a physical robot without pretrained policies.

Chapter 4

Adaptive Arbitration between Model-based and Model-free Control

4.1 Introduction

Recent success in deep RL for continuous control has been dominated by model-free approaches which, unlike model-based approaches, do not suffer from representational limitations, when the model representation imperfectly captures the environment dynamics, and model errors inevitable in complex domains. However, they require lots of experience compared to model-based approaches that are typically sample-efficient. Combining model-free and model-based learning systems, therefore, allows learning complex control policies while also improving the sample efficiency. It is also consistent with a large body of behavioral and neural evidence showing that model-free and model-based learning systems both have an active role in human motor learning (Haith and Krakauer, 2013; Lee et al., 2014).

However, dual-system approaches fail to consider the reliability of the learned model when it is applied to make multiple-step predictions, resulting in a compounding of prediction errors and performance degradation. In an effort to reduce the effect of the compounding prediction error of the learned model during planning, the work of (Talvitie, 2017) proposes a model-based RL algorithm where the model is trained via hallucinated replay to predict the next world state, given its own predictions as input, continually correcting itself. The algorithm has a theoretical guarantee on the error bound of the value of the target policy, but is limited to deterministic environments. More recent work has shown that forcing the latent variables to predict the long-term future using an auxiliary cost during model training makes planning in the latent space involve less prediction errors (Ke et al., 2019).

In this chapter, we will discuss in more detail dual-system motor learning and how it is applied to continuous control. We will then introduce the Curious Meta-Controller (CMC) approach that integrates the two learning systems in an adaptive, reliable and sample-efficient manner. CMC adaptively arbitrates between model-based planning and model-free RL. The arbitration is controlled online via a curiosity signal based on the learning progress of an evolving dynamics model. In contrast to previous dual-system approaches, our approach takes the reliability of the learned model into account before using it for planning. CMC can be combined with any off-policy RL algorithm with minimal changes and is in line with findings from neuroscience on the dual-system approach to human decision-making. We will evaluate popular deep continuous-action RL algorithms with and without CMC and show that CMC improves the sample efficiency and achieves better performance.

4.2 Dual-System Motor Learning

Motor behavior can be divided into habitual behavior obtained by model-free learning and goal-directed behavior obtained by model-based learning (Daw et al., 2005). Several hypotheses were proposed to explain how the human brain arbitrates between model-based and model-free learning systems. For instance, Cushman and Morris (2015) argue that when performing a sequential decision-making task, humans use the model-free system to habitually select goals and then the model-based system to generate a plan to achieve a selected goal. Another study proposes a contrasting hypothesis called “plan-until-habit”, in which planning is first performed by simulating the world up to a certain depth that decreases with increased time pressure and then model-free action values are exploited (Keramati et al., 2016). While this study attributes the change in behavior between model-based and model-free control to the availability of cognitive resources, particularly time, other studies have found that the behavior instead changes according to the expected reward regardless of resource availability (Kool et al., 2018; Boureau et al., 2015). Kool et al. (2018) support the latter by providing behavioral evidence that people with a perfect transition model of the task and an extended response deadline exerted less model-based control when its expected reward advantage was lower than the cognitive cost involved. This finding was interpreted by suggesting that the brain estimates the value of using each control system but reduces that of the model-based system in proportion to its increased cognitive cost. Similarly, Boureau et al. (2015) state that meta-decisions including arbitration between model-based and model-free control are governed by a cost-benefit trade-off in which the brain constantly generates rough estimates of the costs and benefits of allocating cognitive resources for model-based control. The average reward rate, which is the reward expected for temporally allocating a particular resource, and the controllability, which measures how advantageous a carefully considered decision is over a fast habitual one in terms of rewards collected, are proposed as estimates for the opportunity cost and benefit respectively. The willingness to exert model-based control thus increases proportionally to how much larger the reward obtained by controllability is compared to the average reward rate. The authors note, however, that modifying these estimates by including other decision variables like the uncertainty about action outcomes might account for meta-decisions in specific behavioral contexts like exploration.

Haith and Krakauer (2013) review the behavioral evidence for the existence of each of the model-based and model-free mechanisms of motor learning in humans and argue that both are employed in parallel by the motor system for movement control. They point out that while the two learning systems generate their own estimate of the value of a given action at a given state, the decision which action to take is made primarily based on the reliability of each of these two estimates. Imperfect predictions of an internal forward model limit the reliability of model-based learning and, hence, in the later stages of learning, after extensive experience, model-free learning becomes more reliable, as the authors indicate. Another study provides neural evidence that the human brain encodes the reliability of model-based and model-free learning systems based on their prediction errors in the lateral prefrontal and frontopolar cortex and uses the reliability signals to dynamically arbitrate behavioral control between the two systems (Lee et al., 2014). The arbitration model the study proposes combines model-based and model-free value signals, weighted by the degree of reliability of each system, and uses this integrated value signal for guiding behavior. To account for the cognitive complexity involved, the arbitrator incorporates a bias toward the less cognitively demanding model-free control. The arbitration between different learning systems was also found to drive human strategy selection where the goal is to learn when to use which strategy (Lieder and Griffiths, 2015). The proposed context-sensitive strategy selection approach, which assumes a mental model predicting each strategy's accuracy and execution time from features of the current situation, was found to better explain how people adaptively choose strategies than previous accounts. However, it is based on choosing the strategy with the best predicted speed-accuracy trade-off rather than choosing the most reliable strategy.

In contrast to previous works suggesting strict neural and behavioral division between model-based and model-free learning systems, Russek et al. (2017) propose a computational framework where the two systems are tightly coupled, motivated by evidence supporting a role for dopamine in model-based learning besides its well-established role in model-free learning. In the framework, action values are estimated by applying model-free temporal difference (TD) learning to successor representations (SR), which are the expected future state occupancies. This was found to explain the involvement of dopamine in model-based learning, since the TD error is a reward prediction error thought to be mediated by phasic dopamine and SR is a predictive representation capturing knowledge of the transition model. Although the presented framework gives a neurally plausible computational account of the interaction between the two learning systems, it does not answer the question of how the brain prioritizes computations and arbitrates control between learning systems, as concluded by the authors.

While these studies provide strong evidence for the dual-system approach to human motor learning that is distinguishable neurally and behaviorally and can be used in robot motor learning, they almost always assume a perfect internal model of the task. To relax this assumption, an intrinsic measure of the reliability in model predictions needs to be incorporated into the behavioral control system. This is most likely to guide the behavior to improve the learned model and eventually lead to better task performance.

4.3 Dual-System Deep RL for Continuous Control

Deep RL approaches are broadly classified into model-based and model-free ones. Model-based approaches facilitate transfer of learning across tasks, since a model learned in the context of a task can be directly used to compute an appropriate control policy for a new task. They are also typically sample-efficient in that they allow for generating synthetic experiences by making predictions about the future. On the other hand, model-free approaches are free from representational limitations that would prevent the convergence to a desired behavior if the model representation is insufficient to perfectly capture the environment dynamics. However, they require a lot of experience and hence have high sample complexity. This has motivated several works to address the problem of how to combine the benefits of model-based and model-free methods.

Initializing the neural network policy of a model-free learner using rollouts of a model-based controller followed by model-free fine-tuning of the policy was found to lead to a higher sample efficiency compared to pure model-free learning with random policy initialization (Nagabandi et al., 2018). The model-based controller used is based on random-sampling where several randomly generated action sequences are fed to the model and then the sequence with the highest expected reward is chosen. This limits the effectivity of the approach to low-dimensional action spaces and short horizons. In a different work, Feinberg et al. (2018) decompose action-value estimation into two parts: one contains the sum of future rewards predicted by a learned model over a limited horizon and one contains the cached model-free estimate of the long-term reward computed at the end of the horizon. While the method is shown to boost the sample efficiency, it assumes perfect model predictions for a fixed horizon, which is a strong assumption, because, in practice, the model generates noisy data early in learning, and a measure of model reliability is therefore needed.

Other works used information about the future provided by a trained world model as input to the model-free learner to improve its decisions (Racanière et al., 2017; Ha and Schmidhuber, 2018). In (Racanière et al., 2017), imagined trajectories generated by a model are processed by a recurrent neural network that outputs a rollout encoding for each trajectory. The encodings are concatenated and used as additional context for the model-free learner’s value and policy networks. Rather than training a feedforward model, Ha and Schmidhuber (2018) train a recurrent world model on random environment rollouts and use the hidden state of the trained model along with a learned abstract state representation as input to a model-free controller. The proposed approach achieves state-of-the-art performance on an image-based car racing task. However, these works employ pretrained world models and abstract representations with the risk of encoding task-irrelevant features. Instead, Francois-Lavet et al. (2019) propose training the world model and an abstract state representation that minimizes both model-free and model-based losses during task learning. The abstract state is the input to both the model-free Q-network predicting action values and the world model predicting the next states and rewards. Planning is done by performing one fixed-depth rollout of the model for each possible action at the current state and then taking the first action of the rollout with the highest overall estimated value. The approach has two major drawbacks: first, the complexity of planning, which is performed at each time step, grows exponentially with the number of possible actions; second, if the model is inaccurate, as is the case in complex domains, a large fixed planning depth leads to a compounding of prediction errors that eventually impairs task performance.

Learning a state representation and a dynamics model that make gradient-based planning generate actions that mimic expert demonstrations has been found to yield successful action plans (Srinivas et al., 2018). The distance in the learned representation space between the predicted terminal state and the goal state is shown to provide a useful reward for model-free RL in tasks with image-based goals. The approach however requires expert demonstrations to be available during training. In (Fard and Trappenberg, 2018), a control architecture is proposed that includes an arbitrator used to switch between habitual and planning systems by choosing between an action predicted by an actor of a model-free actor-critic system and that predicted by an inverse dynamics model. The arbitration is managed by the reward prediction error and favors the actor's prediction if the error at the previous time step is below a predefined threshold. The approach does not address model imperfection and is applied to a significantly low-dimensional state space. As opposed to explicitly learning a dynamics model, Pong et al. (2018) propose a type of goal-conditioned value function called Temporal Difference Model (TDM) that implicitly learns a dynamics model and uses it for optimal control. In their approach, transitions collected off-policy are sampled from a replay buffer and relabeled with new, randomly sampled goal states and time horizons which the TDM uses as input along with the state-action pair. The TDM is learned model-free and updated to be the negative distance between the newly visited and goal states if the horizon is zero or, otherwise, the TDM value after decrementing the horizon and advancing the state. The information the TDM provides on the closeness to the goal after a given number of actions makes it resemble a model. Despite achieving high sample efficiency by relabeling collected transitions with several goals and horizons, the approach is not applicable to domains where the goal representation is embedded in a raw-pixel observation and thus cannot be given as a separate input channel.

4.4 Curious Meta-Controller

In this section, we present our Curious Meta-Controller (CMC) approach for adaptive arbitration between model-free and model-based control. CMC consists of model-free and model-based control systems and a meta-controller deciding on which of the two systems to query for an action at each time step. We first discuss in more detail each of the two systems and then show how the reliability in model predictions is used to adaptively guide meta-decisions and provide an intrinsic reward to improve the model. Our objective is to train a policy neural network representing the desired control behavior more efficiently than when following a pure model-based or model-free approach.

4.4.1 Model-free Control with Off-policy Actor-Critic

To train a model-free controller from experience, we consider actor-critic methods which are well suited for continuous control by learning simultaneously a value function and a policy function (Section 1.2.1). We are particularly interested in off-policy actor-critic methods, since they allow for learning from actions coming from different systems, such as a model-based controller. Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) is a state-of-the-art off-policy actor-critic method that we use in our approach along with an off-policy variant of Continuous Actor-Critic Learning Automaton (CACLA) (Van Hasselt, 2012).

Our actor-critic architecture is shown in Figure 4.1. The architecture consists of a jointly trained critic-autoencoder network (Figure 4.1(a)) and an actor network (Figure 4.1(b)). The latent state representation ϕ_s, which is the output of the convolutional encoder $f(s|\omega)$, is learned to be state discriminator and value predictor by jointly optimizing the combined reconstruction and value prediction loss:

$$\mathcal{L}_{combined}(\omega, \theta^Q, \tilde{\omega}) = \lambda_{rec}\mathcal{L}_{rec}(\tilde{\omega}, \omega) + \lambda_Q\mathcal{L}_Q(\theta^Q, \omega), \tag{4.1}$$

where $\mathcal{L}_{rec}(\tilde{\omega}, \omega) = \left\| g\left(\phi_{s_t}|\tilde{\omega}\right) - s_t \right\|_2^2$ is the reconstruction loss between the decoder's output $g(\phi_{s_t}|\tilde{\omega})$ and the original input s_t, $\mathcal{L}_Q(\theta^Q, \omega) = \left(y_t - Q(s_t, a_t|\omega, \theta^Q)\right)^2$ is the value

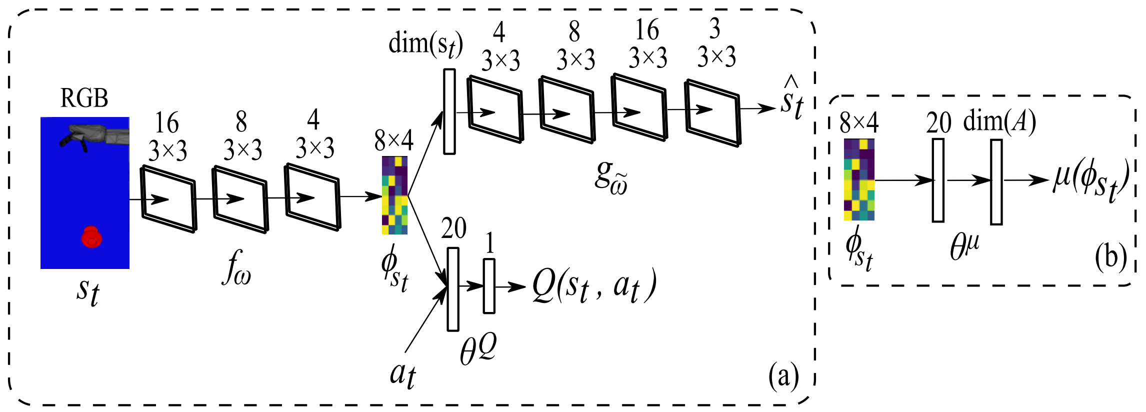


Figure 4.1: Model-free control system: (a) Critic-autoencoder network consisting of a fully convolutional encoder f_ω that takes in a raw image s_t, a fully convolutional decoder $g_{\tilde{\omega}}$ that computes a reconstruction $\hat{s}_t$, and a critic Q that estimates the Q-value given s_t and a_t; (b) Actor network taking in the latent state representation ϕ_{s_t}, which is jointly trained to minimize the reconstruction and value prediction losses, and generating a control action $\mu(\phi_{s_t})$ with a dimensionality of dim(A), where A is the action space.

prediction loss between the expected value $Q(s_t, a_t|\omega, \theta^Q)$ and the target value $y_t = r_t + \gamma Q'\left(s_{t+1}, \mu'\left(\phi_{s_{t+1}}|\theta^{\mu'}\right)|\omega', \theta^{Q'}\right)$, $Q'(\cdot,\cdot\,|\omega', \theta^{Q'})$ and $\mu'(\cdot\,|\theta^{\mu'})$ are the target critic and actor networks respectively, and λ_{rec} and λ_Q are weighting coefficients on the individual loss components. This jointly learned latent representation captures task-relevant information sufficient to reconstruct the original input and recognize rewarding states and is therefore used as a direct input to the actor network, as shown in Figure 4.1(b). The actor is trained according to the chosen actor-critic algorithm. DDPG updates the actor parameters by minibatch gradient ascent on the Q-function:

$$\theta^\mu_{t+1} \leftarrow \theta^\mu_t + \frac{\beta}{n}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)}\ \nabla_{\theta^\mu}\mu\left(\phi_s|\theta^\mu_t\right)|_{s=s_i}, \tag{4.2}$$

where n is the minibatch size and β is the gradient step size ($0 < \beta \leq 1$), whereas off-policy CACLA updates the actor only when the advantage δ of taking an action a is positive by gradient ascent on the loss $\mathbb{E}_{(s,a,r,s')\sim B, \delta>0}\left[\left(a - \mu\left(\phi_s|\theta^\mu\right)\right)^2\right]$:

$$\theta^\mu_{t+1} \leftarrow \theta^\mu_t + \beta\nabla_{\theta^\mu_t}\ \mathbb{E}_{(s,a,r,s')\sim B, \delta>0}\left[\left(a - \mu\left(\phi_s|\theta^\mu_t\right)\right)^2\right], \tag{4.3}$$

where (s, a, r, s') are experience tuples sampled from a replay buffer B and $\delta = r + \gamma Q'(s', \mu(s'|\theta^{\mu'})|\theta^{Q'}) - Q(s, \mu(s|\theta^\mu)|\theta^Q)$ is the action advantage, representing how better the observed value of taking a is than the expected value $Q(s, \mu(s|\theta^\mu)|\theta^Q)$. This moves the actor's output towards an action a that has a positive advantage.

4.4.2 Model-based Control with MPC

In our proposed approach, a predictive model of the world dynamics is learned simultaneously with the task. Instead of learning the model at pixel-level, which is noise sensitive and infeasible in practice, the model is learned in the space of jointly trained latent representations (Figure 4.1(a)). This also ensures that the model is learned on task-relevant latent representations, as opposed to representations learned only to minimize the pixel-level reconstruction error of an autoencoder, which includes no information on what features are useful for the task. The latent-space world model predicts the next latent state representation and extrinsic reward given the current representation and action and is trained to minimize the loss:

$$\mathcal{L}_{model}(\theta^{\mathcal{M}}, \theta^{\mathcal{R}}) = \left\| \mathcal{M}\left(\phi_{s_t}, a_t | \theta^{\mathcal{M}}\right) - \phi_{s_{t+1}} \right\|_2^2 + \left\| \mathcal{R}\left(\phi_{s_t}, a_t | \theta^{\mathcal{R}}\right) - r_t^{ext} \right\|_2^2, \quad (4.4)$$

where r_t^{ext} is the extrinsic reward, $\mathcal{M}(\cdot,\cdot\,|\theta^{\mathcal{M}})$ and $\mathcal{R}(\cdot,\cdot\,|\theta^{\mathcal{R}})$ are two feedforward neural networks for predicting the next latent state representation and immediate extrinsic reward respectively (the two loss components are normalized to $[0,1]$).

To perform planning with a learned latent-space world model, we use model predictive control (MPC). In MPC, the model is rolled out multiple time steps into the future, starting from an initial state and provided with an action sequence. An objective function is measured at each time step, and then by backpropagation through time and gradient descent, an action sequence that optimizes the objective is computed. Only the first action of the optimal action sequence is applied, and the process is repeated at the next time step with the updated state information in closed loop. In our approach, the initial action sequence is provided by the model-free RL actor (Section 4.4.1) at the initial and subsequent model-generated latent states and is optimized with MPC over a time horizon H by minimizing the loss:

$$\mathcal{L}_{plan}(\hat{a}) = \left(R^* - \sum_{h=t}^{t+H-1} \hat{r}_h \right)^2, \quad (4.5)$$

where $\hat{r}_h = \mathcal{R}\left(\hat{\phi}_{s_h}, \hat{a}_h | \theta^{\mathcal{R}}\right)$ is the predicted reward at time step h, $\hat{\phi}_{s_h} = \mathcal{M}\left(\hat{\phi}_{s_{h-1}}, \hat{a}_{h-1} | \theta^{\mathcal{M}}\right)$ is the latent state predicted by $\mathcal{M}$, $\hat{a}_h = \mu\left(\hat{\phi}_{s_h} | \theta^{\mu}\right)$ is the actor's output, and R^* is the desired return. We perform K gradient descent steps on $\mathcal{L}_{plan}$ (Equation 4.5) with respect to each individual action in the initial plan:

$$\hat{a}_{t:t+H-1}^{(i+1)} = \hat{a}_{t:t+H-1}^{(i)} - \alpha_{plan} \nabla_{\hat{a}_{t:t+H-1}^{(i)}} \mathcal{L}_{plan}^{(i)}, \quad (4.6)$$

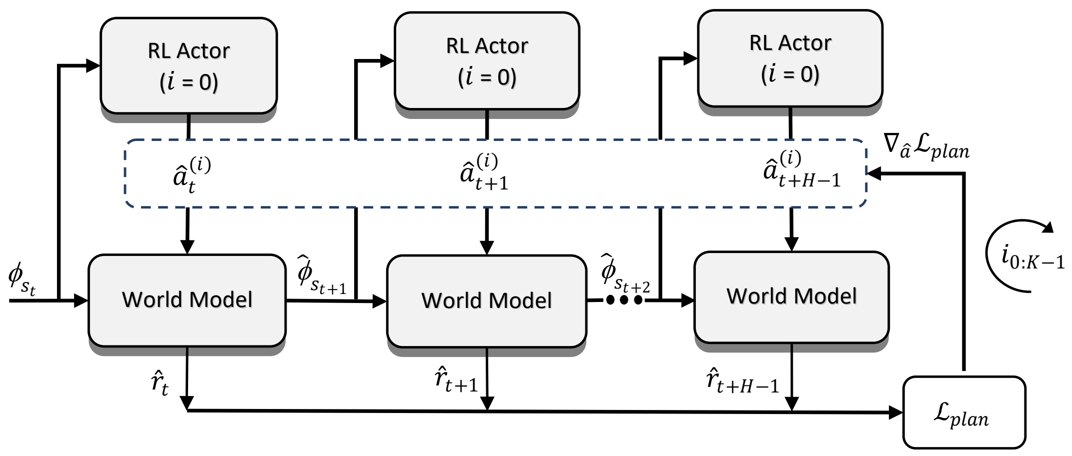


Figure 4.2: Model-based control system: After observing the latent state ϕ_{s_t}, the world is simulated *H* time steps into the future using the learned world model and an initial action sequence $\hat{a}_{t:t+H-1}^{(0)}$ proposed by the RL actor, resulting in a sequence of model-generated latent states $\hat{\phi}_{s_{t+1:t+H-1}}$ and rewards $\hat{r}_{t:t+H-1}$. The objective $\mathcal{L}_{plan}(\hat{a})$ is then measured and optimized by performing backpropagation and *K* steps of gradient descent. The first action of the optimal plan $\hat{a}_0^{(k-1)}$ is applied to the environment and the optimization process is repeated at the next time step.

where α_{plan} is the learning rate for plan optimization. This results in an optimal plan $\hat{a}^{(K-1)}$ whose first action only is executed in the environment. Figure 4.2 shows one iteration of this optimization process in which an action plan that optimizes the objective $\mathcal{L}_{plan}(\hat{a})$ given the model is inferred.

4.4.3 Meta-Decision Making

During task learning, we maintain a moving window average of the prediction error of the latent-space dynamics model:

$$\langle e_t^{prd} \rangle = \frac{1}{\sigma} \sum_{i=t-\sigma+1}^{t} e_i^{prd} \,\Big|_{e_i^{prd} = \left\| \mathcal{M}(\phi_{s_i}, a_i | \theta^{\mathcal{M}}) - \phi_{s_{i+1}} \right\|_2^2 + \left\| \mathcal{R}(\phi_{s_i}, a_i | \theta^{\mathcal{R}}) - r_i^{ext} \right\|_2^2} \tag{4.7}$$

where σ is a time window and e_i^{prd} is the model prediction error at time step *i*. We also monitor the performance improvement in prediction over time by continually measuring the model learning progress:

$$LP_t = \langle e_{t-\mathcal{W}}^{prd} \rangle - \langle e_t^{prd} \rangle \tag{4.8}$$

where $\mathcal{W}$ is a time window. The learning progress is used to derive an intrinsic reward $r_t^{int} = -LP_t$, which represents the agent's curiosity to improve its knowledge of the world, encouraging actions that yield data that improves the model. This is achieved by combining the extrinsic and intrinsic rewards:

$$r_t = r_t^{ext} + \frac{r_t^{int}}{1+ D\cdot t} \tag{4.9}$$

where r_t^{ext} is the extrinsic reward and $D > 0$ is a decay constant used for annealing the intrinsic reward magnitude over time, since the uncertainty in the world dynamics is reduced with exploration. The combined reward r_t is then used to update the critic. The learning progress is also used as an unbiased reliability estimator that underlies meta-decisions.

At each time step of the learning process, a standard model-free off-policy actor-critic method suggests an exploratory action that arbitrarily deviates from the actor's output in the hope to find and learn better actions. Similarly, a model-based planning method finds an optimal action plan by simulating the world using a predictive model with the risk of employing highly imperfect predictions. CMC presents an integrated more efficient exploration method that adaptively decides which of the model-free and model-based control systems to query at each time step. This meta-decision is based on the learning progress of a latent dynamics model. If the learning progress at the previous time step LP_{t-1} is positive, which indicates high prediction reliability, CMC queries the model-based control system for an optimal action (using an initial action sequence suggested by the model-free RL actor), providing a promising alternative to any arbitrary action. Otherwise, a negative learning progress indicates low prediction reliability which increases the level of curiosity of the agent, motivating the selection of actions that improve the model. Since this curiosity is represented by the intrinsic reward used in combination with the extrinsic reward to train the critic of the model-free system, CMC queries the model-free system's actor for an optimal action. This action helps improve the learned model so that future planning with the model will become more accurate.

In our approach, both the model-based and the model-free control systems are mutually improving, since the model-free system provides the model-based system with a good initial action sequence and the model-based system provides the model-free system with a more informed exploratory action. Figure 4.3 shows CMC with its two mutually improving components interacting with the world. The learning algorithm is summarized in Algorithm 3.

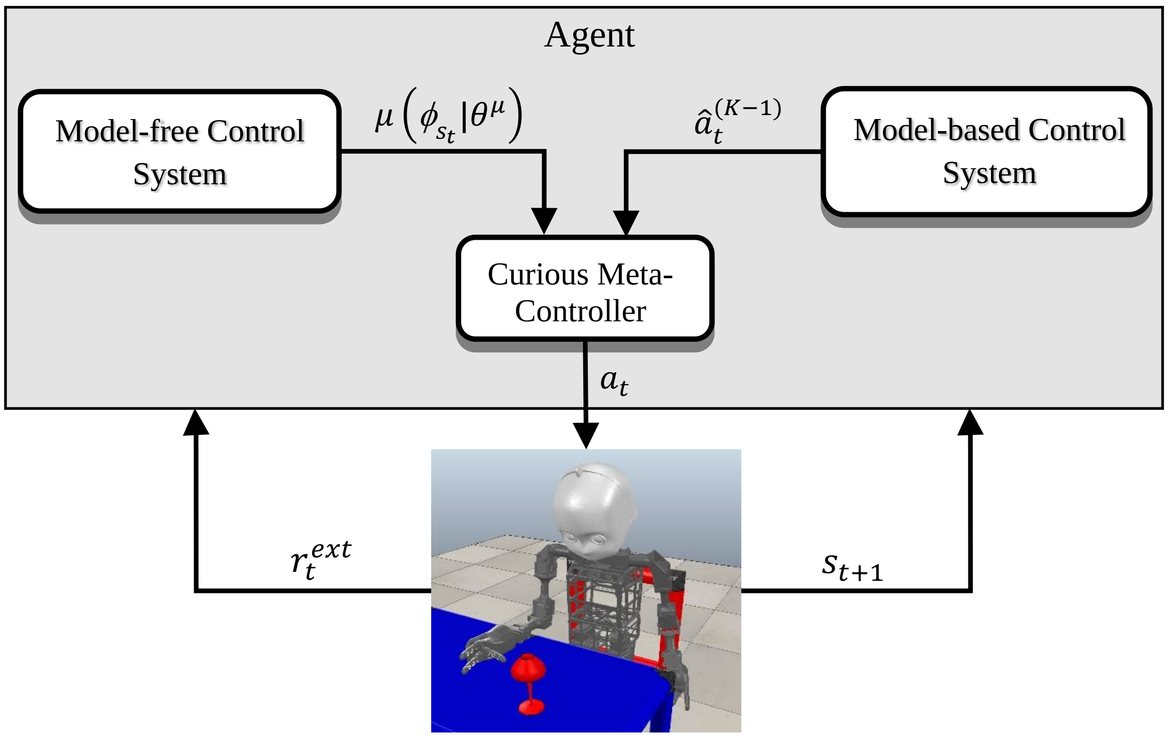

Agent
Model-free Control System
$\mu\left(\phi_{s_t}|\theta^\mu\right)$
$\hat{a}_t^{(K-1)}$
Model-based Control System
Curious Meta-Controller
a_t
r_t^{ext}
s_{t+1}

Figure 4.3: Curious Meta-Controller (CMC): At each time step t, CMC uses the adaptive learning progress LP_{t-1} to decide which controller to query for an action. If LP_{t-1} is positive, CMC queries the model-based control system which then performs planning in the learned latent space. After K optimization iterations performed on an initial action sequence from the model-free RL actor, the optimal plan's first action $\hat{a}_t^{(K-1)}$ is applied to the environment with exploration noise. If LP_{t-1} is negative, CMC queries the actor neural network of the model-free control system for its estimate of the optimal action $\mu\left(\phi_{s_t}|\theta^\mu\right)$ which is then applied to the environment with exploration noise.

4.5 Experiments

We evaluate CMC when used with different off-policy actor-critic methods on learning continuous control tasks from raw pixels. In all experiments, we use the learning architecture shown in Figure 4.1, with the number and size of convolutional filters placed above the corresponding layers, for approximating the policy and Q-functions. All convolutional layers are zero-padded, have stride 1, and use ReLU activations. All dense layers use ReLU activations except for the actor's and critic's output layers that use a tanh and a linear activation respectively. The target networks' update rate τ is 1e-3. The loss weighting constants λ_{rec} and λ_Q are set to 0.1 and 1 respectively. The dynamics model is a feedforward neural network with three dense layers: one hidden layer of 64 tanh units and two output layers for predicting the next latent state and reward with 32

Algorithm 3 Curious Meta-Controller (CMC)

```
1:  **Input:** Planning horizon $H$, no. of plan optimization iterations $K$, episode length $T$,
           no. of episodes $E$, decay constant $D$
2:  **Given:** an off-policy actor-critic method $\mathbb{AC}$
3:  Initialize actor and critic-autoencoder network parameters $\{\omega, \widetilde{\omega}, \theta^{Q}, \theta^{\mu}, \omega', \theta^{Q'}, \theta^{\mu'}\}$
4:  Initialize model network parameters $\{\theta^{\mathcal{M}}, \theta^{\mathcal{R}}\}$
5:  Initialize replay buffer $B$
6:  **for** $episode = 1$ to $E$ **do**
7:    Sample initial state $s_1$
8:    **for** $t = 1$ to $T$ **do**
9:      Compute latent state encoding $\phi_{s_t} = f(s_t|\omega)$
10:     **if** $LP_{t-1} \geq 0$ **then**
11:       Query model-based control system with time horizon $H$ (see Section 4.4.2)
12:       $a_t \leftarrow \hat{a}_t^{(K-1)}$: $\hat{a}_t^{(K-1)}$ is the first action of the optimal plan
13:     **else**
14:       Query model-free control system (see Section 4.4.1)
15:       $a_t \leftarrow \mu\left(\phi_{s_t}|\theta^{\mu}\right)$, where $\mu$ is $\mathbb{AC}$'s actor
16:     **end if**
17:     Add exploration noise $a_t \leftarrow a_t + \mathcal{N}(0, 1)$
18:     Execute $a_t$ and observe $r_t^{ext}$ and $s_{t+1}$
19:     Compute learning progress $LP_t$, following Equation 4.8
20:     Compute intrinsic reward $r_t^{int} = -LP_t$
21:     Compute total reward, following Equation 4.9
22:     Store $(s_t, \phi_{s_t}, a_t, r_t, r_t^{ext}, s_{t+1}, \phi_{s_{t+1}})$ in $B$
23:     Update $\{\theta^{\mathcal{M}_n}, \theta^{\mathcal{R}_n}\}$ using $(\phi_{s_t}, a_t, r_t^{ext}, \phi_{s_{t+1}})$ to minimize $\mathcal{L}_{model}$ (Equation 4.4)
24:     Update $\{\omega, \widetilde{\omega}, \theta^{Q}\}$ on minibatch from $B$ to minimize $\mathcal{L}_{combined}$, (Equation 4.1)
25:     Update $\theta^{\mu}$ on minibatch from $B$ using $\mathbb{AC}$'s actor (see Equations 4.2 and 4.3)
26:     Update target network parameters: $\theta^{Q'} \leftarrow \tau\, \theta^{Q} + (1-\tau)\theta^{Q'}$,
        $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$, $\theta^{\mu'} \leftarrow \tau\, \theta^{\mu} + (1-\tau)\theta^{\mu'}$, with $\tau \ll 1$
27:   **end for**
28: **end for**
```

and 1 linear units respectively. The discount factor γ and decay constant D are set to 0.99 and 0.1 respectively. The time windows σ and $\mathcal{W}$ are set to 40 and 20 respectively. We scale the intrinsic reward to the interval $[-1,1]$. The planning horizon H and the number of plan optimization iterations K are set to 3 and 10 respectively. We train the networks using Adam optimizer (Kingma and Ba, 2014) with learning rate 1e-3 for the critic and the dynamics model and 1e-4 for the actor and a minibatch size of 256. We perform 15 optimization steps to update the critic and actor network parameters and 10 steps for the model network parameters per time step. The replay buffer size is 100K.

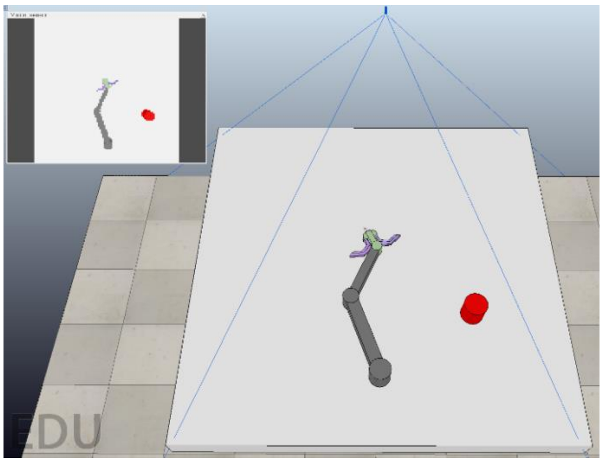


Figure 4.4: V-REP simulation environment for random target reaching. The vision sensor's output (upper-left) is fed as input to the learning algorithms.

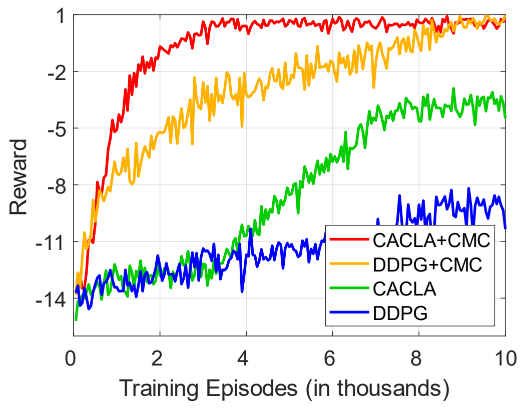
The actor's output is multiplied by a maximum step of 20 units before being sent to the model-based control system or the environment. All hyperparameters were determined empirically through preliminary experiments. We compare the performance of DDPG and off-policy CACLA (hereafter referred to as just 'CACLA') with and without CMC on learning realistic robotic reaching and grasping tasks using V-REP robot simulator (Rohmer et al., 2013). We run the algorithms for 10K episodes and 50 steps per episode on a single Nvidia GTX 1050 Ti 4GB GPU.

4.5.1 Vision-based Robotic Reaching

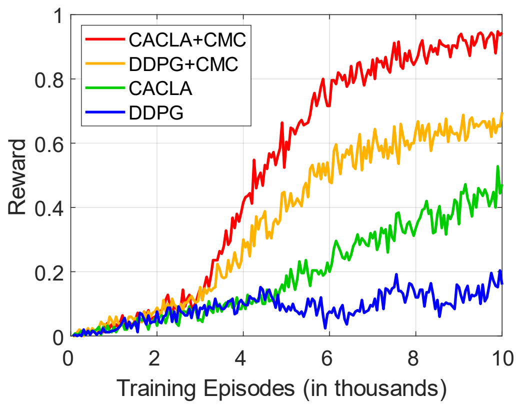
We consider random target reaching using a 3-degree of freedom (DoF) robotic arm with a two-finger gripper and a red cylinder-shaped target object. The 3D robotic environment including the vision sensor's output is shown in Figure 4.4. Real-time 84×84 pixel RGB images from a ceiling vision sensor are used as environment states. The angular range of movement of all arm joints is $\pm\frac{\pi}{2}$. The radius of the target zone centered around the object is one-tenth of the arm's total length and the zone area is approximately 2% of the total area reachable by the arm. The reward function used in the dense reward setting is:

$$r_t^{ext} = \begin{cases} +1 & if\ successful \\ -\|c^t - c^g\| & otherwise \end{cases}$$

where $\|c^t - c^g\|$ is the Euclidian distance between the centers of the target object c^t and the gripper c^g. In the sparse reward setting, the environment returns a reward of one when the target is reached and zero otherwise. In every episode, the position of the target object is initialized randomly within the reachable region.



(a)



(b)

Figure 4.5: Learning curves of DDPG and CACLA with and without CMC on random target reaching from pixel input in two reward settings: (a) dense reward and (b) sparse reward.

Figure 4.5 shows the mean episode extrinsic reward of the algorithms over 5 random seeds. DDPG and CACLA converged to policies of an episode reward of about -10 and -4 respectively in the dense reward setting (Figure 4.5(a)), while their CMC-based counterparts converged to near-optimal policies, with CACLA+CMC reaching a reward peak in less than 4K training episodes. In the challenging sparse reward setting, DDPG showed unstable learning with no improvement in performance and CACLA reached a poor policy of an episode reward of below 0.5, as shown in Figure 4.5(b). Conversely, DDPG+CMC and CACLA+CMC showed a steady increase in the episode reward, converging to 0.69 and 0.94 (i.e. >90% success rate) respectively.

4.5.2 Vision-based Robotic Grasping

In the second experiment, we evaluate the algorithms on vision-based robotic grasping. The need to perform multi-contact motions and to handle rigid-body collisions with a target object renders learning grasping skills more difficult than learning reaching skills. The grasping experiment here is conducted using our Neuro-Inspired COmpanion (NICO) robot (Kerzel et al., 2017). NICO is a child-sized humanoid developed at the Knowledge Technology institute, University of Hamburg, for research on cognitive neurorobotics and on human-robot interaction. Figure 4.6(a) shows the V-REP simulated NICO in a sitting position in front of a table on top of which a red glass is placed and used as a target object for grasping. To prevent self-collisions while also providing a large workspace for learning grasping skills, we consider a control policy that involves the shoulder joint of the right arm and the finger joints of the right hand, as shown in

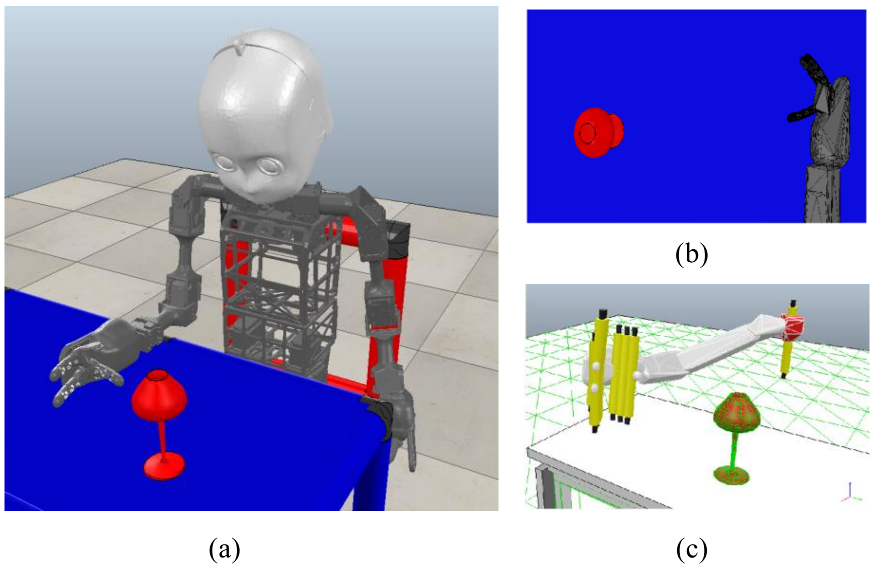


Figure 4.6: Vison-based grasp learning experiment: (a) V-REP simulation environment showing the NICO robot facing a table and a red glass as a grasping target, (b) the sensory input, and (c) motor output (the axes of rotation of the controlled joints are depicted as yellow cylinders).

Figure 4.6(b). NICO's arm has a total of 6 DoFs of which we control one in the shoulder, that has an angular range of movement of $\pm$ 100 degrees. NICO's hand is 11-DoF multi-fingered with 2 index fingers and a thumb, all of which have an angular range of movement of $\pm$ 160 degrees. The robot learns to control 2 DoFs: one for the right shoulder joint and one for the right hand (open/close). The learning algorithms take as input only the 64×32 pixel RGB images obtained from a vision sensor whose output is shown in Figure 4.6(c). The reward function used in the dense reward setting is as follows:

$$r_t^{ext} = \begin{cases} +1 & if\ successful \\ -1 & if\ object\ is\ toppled \\ -\|c^t - c^h\| & otherwise \end{cases}$$

where c^t and c^h are the centers of the target object and the hand respectively. To verify successful grasps, the shoulder joint is moved 20 degrees in the opposite direction to that of the last joint position with the hand closed and the distance $\|c^t - c^h\|$ is measured. If the distance is below a threshold of 0.04 m, the last joint position update is considered successful. Otherwise, the hand is opened and the shoulder joint is moved back to its last position to complete the learning episode. In the sparse reward setting, the environment returns a zero reward for each action that does not result in the object being toppled or grasped. The target object's position is randomly changed to a new graspable position at

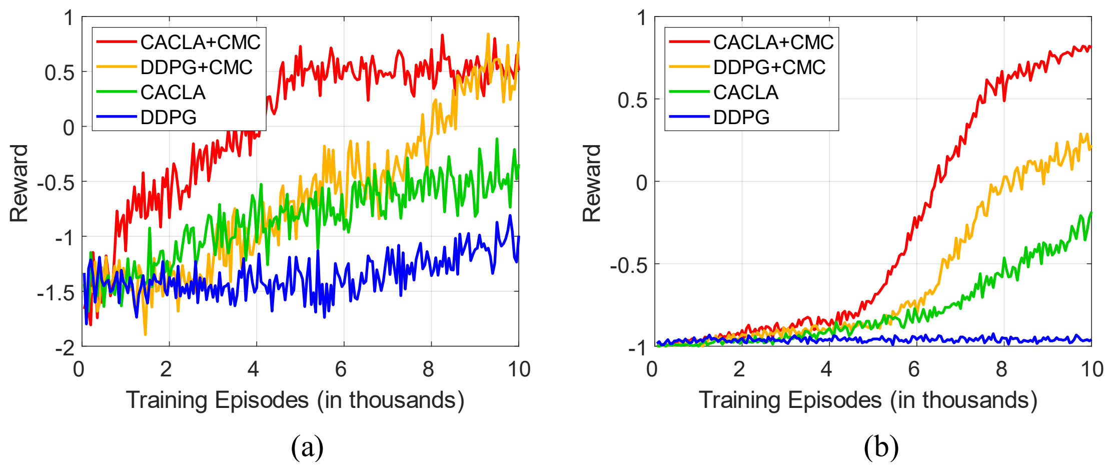
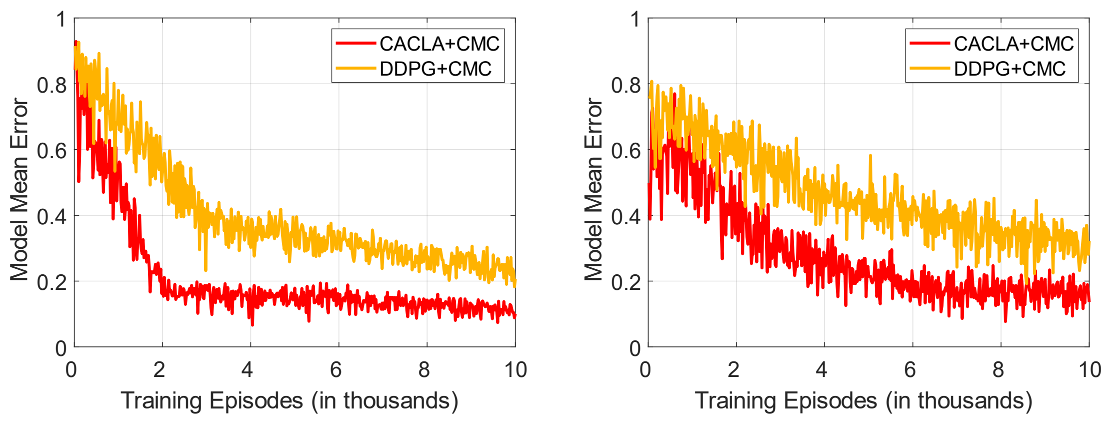


Figure 4.7: Learning curves of DDPG and CACLA with and without CMC on robotic grasping from pixel input in two reward settings: (a) dense reward and (b) sparse reward.

the start of each learning episode. The episode ends when the target object is grasped, toppled, or the maximum episode length T is reached.

The mean episode extrinsic reward of running the algorithms across 5 random seeds is shown in Figure 4.7. All the algorithms showed no considerable performance improvement over the first 2K episodes in the dense reward setting (Figure 4.7(a)). Only CACLA+CMC, however, was able to converge to a policy of mean episode reward of 0.5 in less than 5K episodes, with the other algorithms converging more slowly. The effect of CMC was more evident in the results of the sparse reward setting (Figure 4.7(b)). CACLA+CMC showed a sharp increase in mean episode reward, reaching 0.81 (81% success rate) by the end of learning, while the figure of its CACLA counterpart remained below 0. Likewise, DDPG+CMC's performance gradually improved to a policy of mean episode reward of 0.22, compared to its DDPG counterpart that was unable to improve its performance over the entire learning process.

Figure 4.8 shows the average prediction error of the latent dynamics model over time, normalized to [0,1] and averaged over 5 random seeds in the sparse reward setting. As shown in the figure, the error norm of the model steadily decreased in both robotic reaching and grasping tasks. This shows how the curiosity feedback drives the robot to constantly collect experiences that improve its latent dynamics model and consequently improve the model-based controller's output. The latent dynamics of the reaching environment was learned easier than that of the grasping environment. This is due to a higher accuracy required in the grasping task, which in turn affects the learning speed of the reward prediction part of the model.



(a) (b)

Figure 4.8: The performance of the latent dynamics model of CMC in the sparse reward setting: (a) on the reaching task and (b) on the grasping task.

We also evaluate the effect of using different values of the planning horizon H on learning performance. Figure 4.9 shows the mean episode extrinsic reward of CACLA+CMC on the grasping task with sparse rewards for different planning horizons, averaged over 5 random seeds. Going from a planning horizon of 1 to 3 steps significantly improved the learned policy. For 4-step and 5-step horizons, the performance was already close to that of the 3-step horizon, but with a slight decrease, most likely due to the last model-generated states being outside the reliable sensory region over which the learning progress is computed.

4.6 Conclusion

We introduced Curious Meta-Controller (CMC), a novel curiosity-driven closed-loop controller that adaptively arbitrates between model-free and model-based control systems. The arbitration is determined by an adaptive curiosity signal based on the learning progress of a learned dynamics model in latent space. Unlike previous works, CMC considers the reliability of the model when deciding which of the two control systems to query for an action at each time step and does not require a predefined threshold to arbitrate between them. We evaluated CMC on learning vision-based robotic reaching and grasping in dense and sparse reward environments. The results show that using CMC makes learning pixel-level control policies more sample efficient, particularly in tasks with sparse rewards. CMC can be combined with any off-policy actor-critic method, which we illustrated with DDPG and an off-policy variant of CACLA.

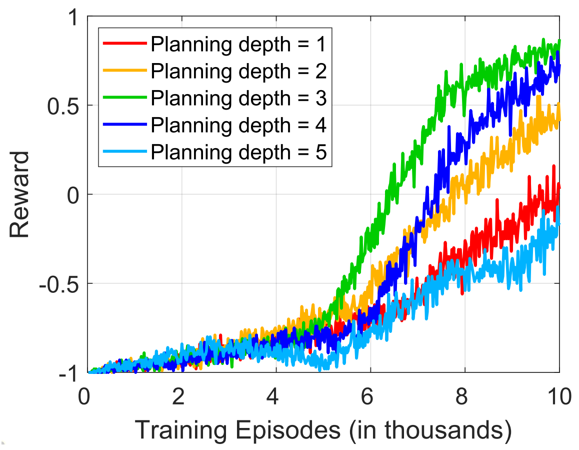


Figure 4.9: Learning curves of CACLA+CMC on the grasping task with sparse rewards for different planning horizons.

Chapter 5

Learning-Adaptive Imagination in Latent Space

5.1 Introduction

Combining model-based and model-free deep RL has shown great promise for improving sample efficiency on complex control tasks while still retaining high performance. Incorporating imagination is a recent effort in this direction inspired by human mental simulation of motor behavior.

Predictive world models are typically known for their ability to boost the sample efficiency of RL methods. In particular, they allow for imagining experiences by making predictions about future states and rewards which can then be used for policy learning, reducing the number of required real experiences of costly agent-environment interactions. Moreover, using features extracted from a recurrent predictive world model as input to an RL agent has been found to achieve state-of-the-art results on challenging RL tasks (Ha and Schmidhuber, 2018). This further confirms the significance of imagination, since the extracted features contain information about the future.

Imagination, defined as mental simulation of motor behavior, is considered strong evidence for cognitive synergy as it requires a combination of different cognitive functions, including abstract perceptual and motor representations, episodic and working memory, and mental manipulation of representations (Moulton and Kosslyn, 2009). This imagination-centered synergy has been further distinguished neurally by examining the different brain regions activated during imagination, including cognitive and motor areas (Case et al., 2015; Ptak et al., 2017) and is a clear example of cognitive development in children where increasingly complex behaviors develop from the recombination of existing, less complex behaviors. Experience imagination is also essential to mental practice which is the cognitive rehearsal of physical skills and found to facilitate skill acquisition (Driskell et al., 1994). Besides, it is estimated that automating imagination has the potential of advancing deep learning beyond finding correlations in data as well as providing a means to broaden the focus of research from problem solving to problem creation through the imagination-supported ability to self-generate goals and pursue them (Mahadevan, 2018). More recently, deep RL methods that employ imagination have been shown to share a number of similarities with human mental simulation, particularly the capacity to build mental models from remembered experiences and using them in decision-making (Hamrick, 2019).

Based on how experience imagination is performed, the approaches for incorporating imagination into deep RL can be divided into two groups: (i) online imagination (Racanière et al., 2017; Nagabandi et al., 2018; Feinberg et al., 2018) and (ii) offline imagination (Gu et al., 2016; Kalweit and Boedecker, 2017). In online imagination approaches, generating imagined trajectories for planning with the world model is done at decision time. Offline imagination approaches, on the other hand, augment the memory of real experiences with model-generated imagined experiences, increasing the amount of data used to train the control policy offline with experience replay. A major issue in these approaches is that they assume a perfect world model. In complex domains, model prediction errors are inevitable and can quickly compound during multi-step action planning, leading to useless long-term predictions. This is one of the reasons model-based RL algorithms have failed to reach the performance of their model-free counterparts in such domains.

In this chapter, we first review the approaches for incorporating imagination into deep RL. We then present the learning-adaptive imagination approach that performs experience imagination in a learned latent space. In our approach, the learned latent space is self-organized into local regions with local world models, and a running average of model prediction error is independently computed for each region. The experience replay buffer is divided into pixel-space and latent-space buffers for storing real and imagined experiences respectively. Imagined rollouts are reliably generated with probability inversely proportional to the average error of the current region, and the imagination depth is adaptively determined by the average error of the traversed regions. we use an intrinsic reward based on the spatially and temporally local learning progress to encourage collecting data that improves future predictions necessary for imagination. Our experiments show that our approach to imagination makes learning robotic grasping from raw pixels more efficient, particularly in sparse reward environments.

5.2 Experience Imagination in Deep RL

Combining model-free and model-based RL is a well-studied problem. One of the earliest works in this direction is Dyna-Q (Sutton, 1990) which learns an action-value function from both real and model-generated experiences. Over the last five years, there has been a growing interest in developing Dyna-like methods in deep RL. For example, Gu et al. (2016) augment the replay buffer of state transitions with imagined on-policy transitions generated under a learned model to speed up model-free RL. They iteratively refit a linear model to recently collected transitions and generate short imagined rollouts from states sampled from these transitions. While they attempt to reduce model bias by sampling from regions where the model has recently been trained, the learned linear model is insufficient to perfectly capture complex environment dynamics and generate imagined rollouts in tasks involving learning from raw pixels, as the authors indicate. In contrast, Kalweit and Boedecker (2017) use imagined transitions for updating the value and policy functions only when the action-value estimates have high uncertainty computed via bootstrap. The approach is shown to improve the efficiency of learning continuous control tasks, but does not consider model prediction errors and requires training of additional critic networks for bootstrap uncertainty estimation.

Racanière et al. (2017) follow a different path by using imagination as a context for model-free value estimation. This is done by encoding rollouts of imagined observations with a recurrent neural network. The encoded rollouts are concatenated and used as an additional input to the value and policy functions. In another work, a model-based controller is used such that it randomly generates a number of candidate action sequences at each time step, simulates the imagined trajectories with a learned model, and executes the first action of the trajectory yielding the highest reward (Nagabandi et al., 2018). The controller is then used to initialize the policy of a model-free RL agent with supervised data by providing it with target actions at some sampled states, which is found to make the chosen model-free RL method more sample-efficient. The use of a model predictive controller based on random sampling, however, limits the application of the approach to low-dimensional action spaces and short planning horizons.

Unlike previous works, Feinberg et al. (2018) decompose value estimate into a part with imagined rewards predicted by a dynamics model over a short horizon and a subsequent part estimated by a model-free critic. Their method, called Model-based Value Expansion (MVE), is shown to boost the sample efficiency of learning, but on control tasks with low-dimensional observations (<20). MVE avoids issues related to learning from data generated with an outdated model by not using an imagination buffer. However, it relies on the strong assumption that the model is accurate over a short, fixed horizon. This is most likely to fail in practice, since the model can generate noisy data early in the learning process while still being trained jointly with the target policy, and a measure of model prediction accuracy becomes necessary.

While these approaches incorporate imagination irrespective of the prediction error of a learned dynamics model, our proposed approach takes this error into account before initiating and while performing imagination by using spatially and temporally local estimates of prediction accuracy, as we will explain in the following section.

5.3 Learning-Adaptive Imagination

Generating imagined data for visuomotor control tasks requires learning perfect world models at the pixel level, which is infeasible in practice. Here, we instead propose to learn the model in latent space. In this section, we first describe our learned latent space and how the space is self-organized into local regions with local world models, based on the Deep ICAC algorithm (see Chapter 3). We show how the spatially and temporally local learning progress is used to compute an intrinsic reward that encourages collecting experience data that improves the model. We then present our imagination approach that generates training sequences of imagined experiences in local regions where the corresponding local models retain high prediction accuracy.

5.3.1 Latent Representation Learning

We build upon the Deep ICAC algorithm which learns an ensemble of local dynamics models and generates an intrinsic reward based on learning progress. A latent representation is learned by jointly minimizing a combined convolutional autoencoder's reconstruction and value prediction loss:

$$\begin{aligned}
\mathcal{L}_{rec}(\widetilde{\omega}, \omega) &= \left\| g\left(\phi_{s_t} | \widetilde{\omega}\right) - s_t \right\|_2^2, \\
\mathcal{L}_{critic}(\theta^V, \omega) &= \left(y_t - V(s_t | \omega, \theta^V)\right)^2, \\
\mathcal{L}_{combined}(\omega, \theta^V, \widetilde{\omega}) &= \lambda_{rec}\mathcal{L}_{rec}(\widetilde{\omega}, \omega) + \lambda_{critic}\mathcal{L}_{critic}(\theta^V, \omega),
\end{aligned} \tag{5.1}$$

where $\phi_{s_t} = f(s_t|\omega)$ is the latent state representation at time step t, $f(\cdot\,|\omega)$ and $g(\cdot\,|\widetilde{\omega})$ are the encoder and decoder networks with parameters ω and $\widetilde{\omega}$ respectively, $y_t = r_t + \gamma V'(s_{t+1}, |\omega', \theta^{V'})$ is the target value with $V'(\cdot\,|\omega', \theta^{V'})$ being the critic target network parametrized by $(\omega', \theta^{Q'})$, and λ_{rec} and λ_{critic} are weighting coefficients on the individual loss components. By sharing the learning parameters between the encoder and critic, the latent representation is learned to be a good state discriminator and value predictor and therefore fed as input to the actor network. The actor parameters θ^μ are updated to bring the output closer to an exploratory action a_t found to lead to higher than expected value (i.e. positive TD error). This is done by minimizing the following loss:

$$\mathcal{L}_{actor}(\theta^\mu) = \left(a_t - \mu\left(\phi_{s_t}|\theta^\mu\right)\right)^2. \tag{5.2}$$

The learning architectures of the critic and actor are shown in Figure 5.1(a) and Figure 5.1 (b) respectively.

5.3.2 Latent-Space Self-Organization

In Deep ICAC, the space of learned latent representations is self-organized during exploration into local regions with local dynamics models with the help of a growing self-organizing network. Particularly, we use the *Instantaneous Topological Map* (ITM) (Jockusch and Ritter, 1999). The ITM is defined by a set of nodes i, each with a weight vector w_i, and a set of edges connecting each node i to its neighbors $N(i)$. It starts with two connected nodes, and when a new stimulus ϕ_s is observed, the following adaptation steps are performed:

1. Matching: Find nearest node n and second nearest node n' to ϕ_s:

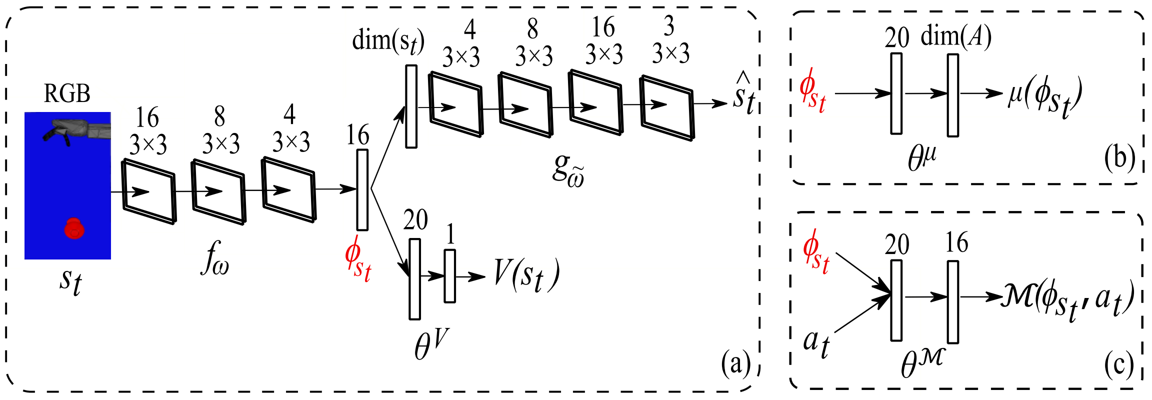


Figure 5.1: Actor-critic and dynamics model architectures: (a) A fully convolutional autoencoder that takes in a raw image s_t and generates a reconstruction $\hat{s}_t$ is jointly trained with the critic and consists of 7 convolutional and 2 dense layers. The number and size of the convolutional filters used are shown above the corresponding layers; (b) The actor is a feedforward network with 2 dense layers whose output dimensionality is dim(A), where A is the action space. It takes as input the 16-D latent representation ϕ_{s_t} trained to minimize the combined critic and reconstruction loss; (c) The dynamics model is a feed-forward network that takes as input the current state's latent representation and the current action. It has one hidden dense layer followed by a dense output layer that outputs a prediction of the latent representation at the next time step.

$$n \leftarrow \arg\min_i \left\|\phi_s - w_i\right\|_2^2, n' \leftarrow \arg\min_{j, j\neq n} \left\|\phi_s - w_j\right\|_2^2.$$

2. Edge adaptation: Create an edge between n and n' if they are not connected. Check, for all nodes m in $N(n)$, if n' lies inside the Thales sphere through m and n (the sphere with diameter $w_m w_n$), $i.e. (w_n - w_{n'}) \cdot (w_m - w_{n'}) < 0$. If true, remove the edge between n and m, and then, if m has no remaining edges, remove m.
3. Node adaptation: If ϕ_s lies outside the Thales sphere through n and n', $i.e. \left(w_n - \phi_s\right) \cdot \left(w_{n'} - \phi_s\right) > 0$, and if $\left\|\phi_s - w_n\right\|_2^2 > e_{max}$, where e_{max} is a given threshold, create a new node v with $w_v = \phi_s$ and an edge with n.

An example of an approximate dynamics model $\mathcal{M}(\cdot,\cdot|\theta^\mathcal{M})$, which predicts the next state encoding given the current action and state encoding and is trained to minimize the loss $\left\|\mathcal{M}\left(\phi_{s_t}, a_t|\theta^\mathcal{M}\right) - \phi_{s_{t+1}}\right\|_2^2$, is shown in Figure 5.1(c). In our approach, in order to generate a complete imagined experience, we additionally learn a latent reward function $\mathcal{R}(\cdot,\cdot|\theta^\mathcal{R})$ which predicts the immediate reward and is trained to minimize the loss $\left\|\mathcal{R}\left(\phi_{s_t}, a_t|\theta^\mathcal{R}\right) - r_t\right\|_2^2$. Each region n of the latent space (node in ITM) is assigned a separate local dynamics model $\mathcal{M}_n$ and reward function $\mathcal{R}_n$. During learning, we maintain a moving average of the combined prediction error of $\mathcal{M}_n$ and $\mathcal{R}_n$ of the region over a window of σ recent predictions:

$$\langle e_{t,n}^{prd} \rangle = \frac{1}{\sigma} \sum_{i=1}^{\sigma} e_i^{prd} \mid_{e_i^{prd} = \left\| \mathcal{M}_n\left(\phi_{s_i}, a_i | \theta^{\mathcal{M}}\right) - \phi_{s_{i+1}} \right\|_2^2 + \left\| \mathcal{R}_n\left(\phi_{s_i}, a_i | \theta^{\mathcal{R}}\right) - r_i \right\|_2^2} \tag{5.3}$$

where e_i^{prd} is the ith prediction error. We also monitor the change of average prediction error over time in each region:

$$LP_{t,n} = \left| \langle e_{t,n}^{prd} \rangle - \langle e_{t-\mathcal{W},n}^{prd} \rangle \right| \tag{5.4}$$

where $\mathcal{W}$ is a time window. This change represents the learning progress (LP) the robot has made or expects to make.

When action a_t is taken at state s_t, the resulting e_t^{prd} associated with the best-matching node *n* of ITM (*w.r.t* ϕ_{s_t}) is measured and the corresponding $\langle e_{t,n}^{prd} \rangle$ and $LP_{t,n}$ are updated. The updated $LP_{t,n}$ is then combined with the perception error $e_t^{per} = \left\| \phi_{s_{t+1}} - w_m \right\|_2^2$, where *m* is the nearest node to $\phi_{s_{t+1}}$, to produce an intrinsic reward signal:

$$r_t^{int} = LP_{t,n} + e_t^{per} \tag{5.5}$$

which encourages actions that maximize learning progress and lead to perceptually novel states. This is achieved by using the combined extrinsic and intrinsic reward to update the critic. The locally trained $\mathcal{M}$ and $\mathcal{R}$ provide informative predictions with accuracy estimated by the locally stored average prediction error that can be taken into consideration when producing imagined rollouts, as explained next.

5.3.3 Latent-Space Experience Imagination

In our approach, we perform imagination in latent space. To facilitate this, we split the replay memory into pixel-space and latent-space replay buffers, B_{pixel} and B_{latent} respectively. B_{pixel} contains transitions T_i^{pixel} of the form (s_i, a_i, r_i, s_{i+1}), while B_{latent} contains transitions T_i^{latent} of the form $\left(\phi_{s_i}, a_i, r_i, \phi_{s_{i+1}}\right)$.

When the best-matching node *n* at time step *t* is identified, we generate an on-policy imagined transition with probability proportional to the current accuracy of $\mathcal{M}_n$ and $\mathcal{R}_n$. This is done by first scaling the average prediction error (Equation 5.3), which is an unbiased estimate of how unreliable the recent local predictions are, to [0,1]. A random

Algorithm 4 Learning-Adaptive (LA-) Imagination (ϕ_{s_t}, n)

```
1:  Input: Max. imagination depth $D^{max}$.
2:  Scale $\langle e_{t,n}^{prd} \rangle$ to [0,1] by its maximum over a window $\mathcal{W}$
3:  $i \leftarrow 0,\ \phi_{s_i} \leftarrow \phi_{s_t},\ \langle e_{i,n}^{prd} \rangle \leftarrow \langle e_{t,n}^{prd} \rangle$
4:  Generate random number $c \sim \mathcal{U}[0,1]$
5:  while ($c < 1 - \langle e_{i,n}^{prd} \rangle$) and ($i \leq D^{max}$) do
6:      Generate imagined transition $T_i^{latent} = \left(\phi_{s_i}, a_i, \hat{r}_i, \hat{\phi}_{s_{i+1}}\right)$ using $\mathcal{M}_n$ and $\mathcal{R}_n$, where
        $a_i \sim \pi(\phi_{s_i})$
7:      Store $T_i^{latent}$ in $B_{latent}$
8:      $\phi_{s_i} \leftarrow \hat{\phi}_{s_{i+1}}$
9:      Find best-matching node n to $\phi_{s_i}$
10:     Scale $\langle e_{i,n}^{prd} \rangle$ to [0,1]
11:     Generate $c \sim \mathcal{U}[0,1]$
12:     $i \leftarrow i + 1$
13: end while
```

number c is then drawn uniformly from [0,1] and an imagined transition is generated if $c < 1 - \langle e_{t,n}^{prd} \rangle$ is satisfied. The generated latent state transition $T_t^{latent} = \left(\phi_{s_t}, a_t, \hat{r}_t, \hat{\phi}_{s_{t+1}}\right)$, where $\hat{r}_t = \mathcal{R}_n\left(\phi_{s_t}, a_t | \theta^{\mathcal{R}}\right)$, $\hat{\phi}_{s_{t+1}} = \mathcal{M}_n\left(\phi_{s_t}, a_t | \theta^{\mathcal{M}}\right)$, and $a_t \sim \pi(\phi_{s_t})$, is added to B_{latent}. The imagined $\hat{\phi}_{s_{t+1}}$ is used to identify the next best-matching node and the imagination process is repeated. The generation of imagined transitions fully adapts to the changes in learning the local $\mathcal{M}$ and $\mathcal{R}$ networks. Similarly, the length of the imagined rollout is adaptively determined by the average prediction error in the traversed latent-space regions and is bounded by a maximum imagination depth D^{max} to limit the computational time. The imagination process is detailed in Algorithm 4.

Both the RL controller and the imagination process in our learning system are mutually improving, since the former is motivated to take actions leading to data that improves future predictions necessary for imagination through using Deep ICAC and the latter augments the available training experiences with imagined experiences in an adaptive manner to improve the sample efficiency of the former. Figure 5.2 shows an overview of our learning system, including the RL actor-critic controller and the latent-space experience imagination process. We summarize the overall procedure in Algorithm 5.

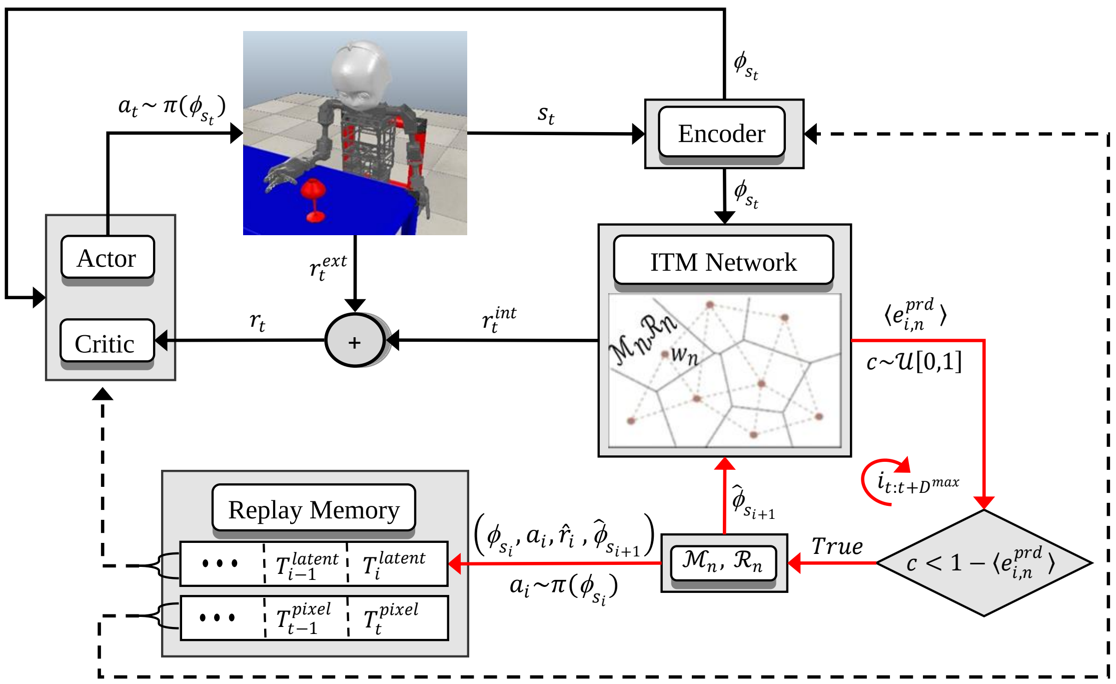

$a_t \sim \pi(\phi_{s_t})$
s_t
ϕ_{s_t}
Encoder
ϕ_{s_t}
Actor
Critic
r_t^{ext}
r_t
+
r_t^{int}
ITM Network
$\mathcal{M}_n \mathcal{R}_n$
w_n
$\langle e_{i,n}^{prd} \rangle$
$c \sim \mathcal{U}[0,1]$
$i_{t:t+D^{max}}$
$\hat{\phi}_{s_{i+1}}$
Replay Memory
T_{i-1}^{latent}
T_i^{latent}
T_{t-1}^{pixel}
T_t^{pixel}
$(\phi_{s_i}, a_i, \hat{r}_i, \hat{\phi}_{s_{i+1}})$
$a_i \sim \pi(\phi_{s_i})$
$\mathcal{M}_n, \mathcal{R}_n$
True
$c < 1 - \langle e_{i,n}^{prd} \rangle$

Figure 5.2: Overview of our system. Solid arrows indicate information flow. Dashed arrows indicate neural network training. At each time step t, the robot observes a new state s_t which is then transformed into latent space with an encoder jointly trained to minimize the combined reconstruction and value prediction loss, as shown in Figure 5.1. The latent encoding ϕ_{s_t} is used to update the growing self-organizing network ITM. The best-matching node n (or the newly created node if the distance is greater than e_{max}) of ITM (w.r.t. ϕ_{s_t}) determines the local dynamics model $\mathcal{M}_n$ and reward function $\mathcal{R}_n$ networks associated with the region of the latent space covered by n. $\mathcal{M}_n$ and $\mathcal{R}_n$ are then updated based on their respective predictions and the observed state transition. The combined prediction error e_t^{prd} is computed and used to update the average $\langle e_{t,n}^{prd} \rangle$. An intrinsic reward r_t^{int} is then derived and combined with the extrinsic reward r_t^{ext} before it is fed to the critic. An on-policy imagined transition, including the predicted next encoding $\hat{\phi}_{s_{t+1}}$ and reward $\hat{r}_t$, is generated with probability inversely proportional to $\langle e_{t,n}^{prd} \rangle$. This imagined transition $T_t^{latent} = \left(\phi_{s_t}, a_t, \hat{r}_t, \hat{\phi}_{s_{t+1}}\right)$ is added to the latent-space buffer of the replay memory. The imagined encoding $\hat{\phi}_{s_{t+1}}$ is used to identify the next best-matching node whose dynamics and reward networks are used to generate the next imagined transition. This imagination process is repeated, adaptively controlled by the probability of generating imagined transitions, up to a maximum imagination depth D^{max}, as shown with red arrows. This is followed by updating the encoder network on a minibatch of transitions from the pixel-space buffer and the actor and critic networks on a minibatch of transitions from the latent-space buffer. Finally, the robot takes a new action sampled from the learned policy with a mean at the actor's output and a new learning cycle starts with a new observed state.

Algorithm 5 Deep ICAC + LA-Imagination

```
 1: Input: Target network's update rate $\tau$, episode length $T$, no. of episodes $E$.
 2: Initialize learning parameters $\{\omega, \tilde{\omega}, \theta^V, \theta^\mu, \omega', \theta^{V'}\}$
 3: Initialize the ITM network
 4: Initialize replay buffers $B_{pixel}$ and $B_{latent}$
 5: for $episode = 1$ to $E$ do
 6:   Sample initial state $s_1$
 7:   for $t = 1$ to $T$ do
 8:     Compute latent state encoding $\phi_{s_t} = f(s_t|\omega)$
 9:     Update the ITM network
10:     Identify the best-matching (or newly created) node $n$
11:     Sample action $a_t \sim \pi$: $\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\left(a_t - \mu(s_t|\theta^\mu)\right)^2/2\sigma^2}$
12:     Execute $a_t$ and observe $r_t^{ext}$ and $s_{t+1}$
13:     Compute intrinsic reward $r_t^{int}$ using Equation 5.5
14:     Compute total reward $r_t = r_t^{ext} + r_t^{int}$
15:     Update $\mathcal{M}_n$ and $\mathcal{R}_n$ networks using the transition $(\phi_{s_t}, a_t, r_t\ \phi_{s_{t+1}})$
16:     Store $(s_t, a_t, r_t, s_{t+1})$ in $B_{pixel}$ and $(\phi_{s_t}, a_t, r_t\ \phi_{s_{t+1}})$ in $B_{latent}$
17:     Call LA-Imagination ($\phi_{s_t}$, $n$) (see Algorithm 4)
18:     Update $\{\omega, \tilde{\omega}\}$ on minibatch from $B_{pixel}$ to minimize $\mathcal{L}_{combined}$ (Equation 5.1)
19:     Update $\{\theta^V, \theta^\mu\}$ on minibatch from $B_{latent}$ to minimize $\mathcal{L}_{critic}$ (Equation 5.1),
        taking $\phi_{s_t}$ as input, and $\mathcal{L}_{actor}$ (Equation 5.2)
20:     Update target network parameters $\theta^{V'} \leftarrow \tau\theta^V + (1-\tau)\theta^{V'}, \omega' \leftarrow \tau\omega + (1-\tau)\omega'$
21:   end for
22: end for
```

5.4 Experiments

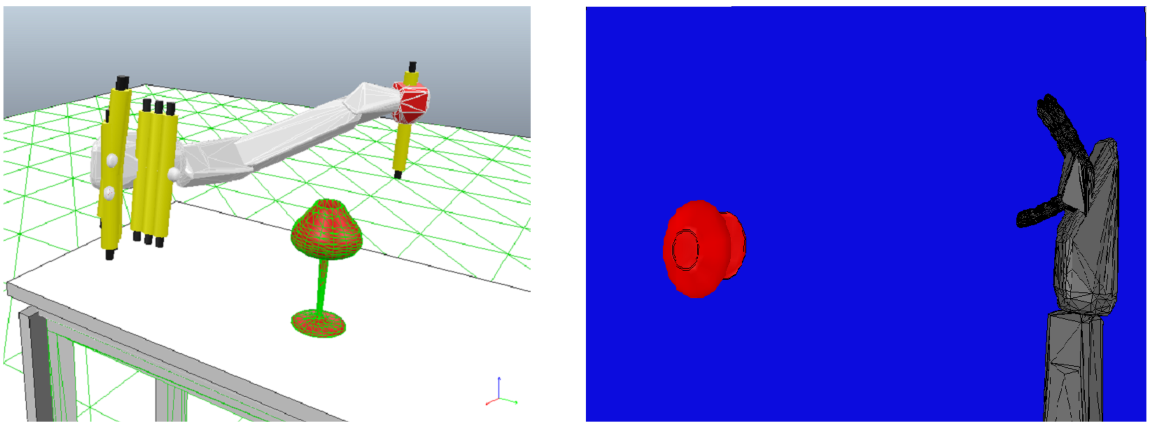
Deep ICAC (Hafez et al., 2019a) has been shown to be more stable and sample-efficient than CACLA (Van Hasselt, 2012) and DDPG (Lillicrap et al., 2016) on visuomotor tasks. Here we show the effect of incorporating learning-adaptive imagination on the sample efficiency. We evaluate our approach on learning vision-based robotic grasping.

Parameters

We use the learning architecture shown in Figure 5.1 for approximating the policy and value functions. All convolutional layers are zero-padded, have stride 1, and use ReLU activations. All dense layers use ReLU activations except for the actor's and critic's output layers that use a tanh and a linear activation respectively. The target network's update rate τ is 1e-3. The loss weighting constants λ_{critic} and λ_{rec} are set to 1 and 0.1 respectively. The functions $\mathcal{M}$ and $\mathcal{R}$ of each region in the latent space are jointly modeled by a feedforward neural network with three dense layers: one hidden layer of 20 tanh units and two output layers for predicting the next latent encoding and immediate reward with 16 and 1 linear units respectively. The discount factor γ is set to 0.99. The time windows σ and $\mathcal{W}$ are set to 40 and 20 respectively. We scale the intrinsic reward to the interval $[-1,1]$. The desired mapping resolution e_{max} which controls the growth of the ITM map and the maximum depth of imagination D^{max} are set to 6.0 and 7 respectively. We train the networks with proportional Prioritized Experience Replay (PER) (Schaul et al., 2016) using Adam optimizer (Kingma and Ba, 2014) and learning rate 1e-3 for the critic, $\mathcal{M}$ and $\mathcal{R}$ functions and 1e-4 for the actor. We use two replay buffers B_{pixel} of size 60K and B_{latent} of size 200K, consuming 40% less memory space than the replay buffer of the Deep ICAC baseline which has a size of 100K, and a minibatch size of 64 sampled by PER. The PER parameters α and β_0 are set to 0.6 and 0.4 respectively. We perform 15 optimization steps on the actor and critic networks and 10 steps on $\mathcal{M}$ and $\mathcal{R}$ per time step. We use a stochastic Gaussian policy with a mean at the actor's output and a standard deviation of 0.35 radians. Actions are capped at 20 units before being sent to the environment. All hyperparameters were determined empirically through preliminary experiments.

Results

We compare the learning performance of Deep ICAC with and without imagination on realistic robotic grasping using V-REP robot simulator (Rohmer et al., 2013). We consider two imagination types: static and learning-adaptive. The former generates imagination rollout of length D^{max} at each time step regardless of prediction errors and the latter is our proposed approach. Grasp learning is a challenging control task due to the need to perform multi-contact motions and handle rigid-body collisions with a target object. The grasping experiment here is conducted on our Neuro-Inspired COmpanion (NICO) robot (Kerzel et al., 2017). NICO is a developmental humanoid built for research on neurorobotics and multimodal interaction. Figure 5.2 (top-left) shows the V-REP simulated NICO in a sitting position facing a table on top of which a red glass is placed and used as a target object for grasping. To avoid self-collisions while still providing a large work space for learning grasping skills, we consider a control policy involving the shoulder joint of the right arm and the finger joints of the right hand, as shown in Figure 5.3(a).



(a) (b)

Figure 5.3: (a) Motor output and (b) sensory input for the grasp learning task. The axes of rotation of the controlled joints are depicted as yellow cylinders in (a).

NICO's arm has a total of 6-Degree of Freedom (DoF) of which we control one in the shoulder with an angular range of movement of $\pm$ 100 degrees. NICO's hand is 11-DoF multi-fingered with 2 index fingers and a thumb, all having an angular range of movement of $\pm$ 160 degrees. NCO learns to control 2 DoFs: one for the right shoulder and one for the right hand (open/close). Each algorithm takes as its only input the 64×32 pixel RGB image from a vision sensor whose output is shown in Figure 5.3(b). The reward function used is as follows:

$$r_t^{ext} = \begin{cases} +10 & if\ successful \\ -10 & if\ object\ is\ toppled \\ 0 & otherwise \end{cases}$$

To verify whether an attempted grasp was successful, the hand is closed and the shoulder joint is rotated 20 degrees in the opposite direction to that of its newly attained position. The distance between the target object's and the hand's centers is then checked whether it is below a threshold of 0.04 m. If yes, the last joint position update is deemed successful. Otherwise, the hand is opened and the shoulder joint is brought back to its last position and the learning continues.

We run the algorithms on a single Nvidia GTX 1050 Ti GPU for 10K episodes and 50 steps per episode and with the target object's position randomly changing to a new graspable position at the start of each episode. The episode ends when the object is grasped, toppled, or the maximum episode length is reached. The average training time (hours) per run is 29.3$\pm$4.1 over a total of 15 runs (5 for each algorithm). Figure 5.4 shows the mean episode extrinsic reward of running the algorithms over five random seeds. Simply using imagination irrespective of state and reward prediction accuracy

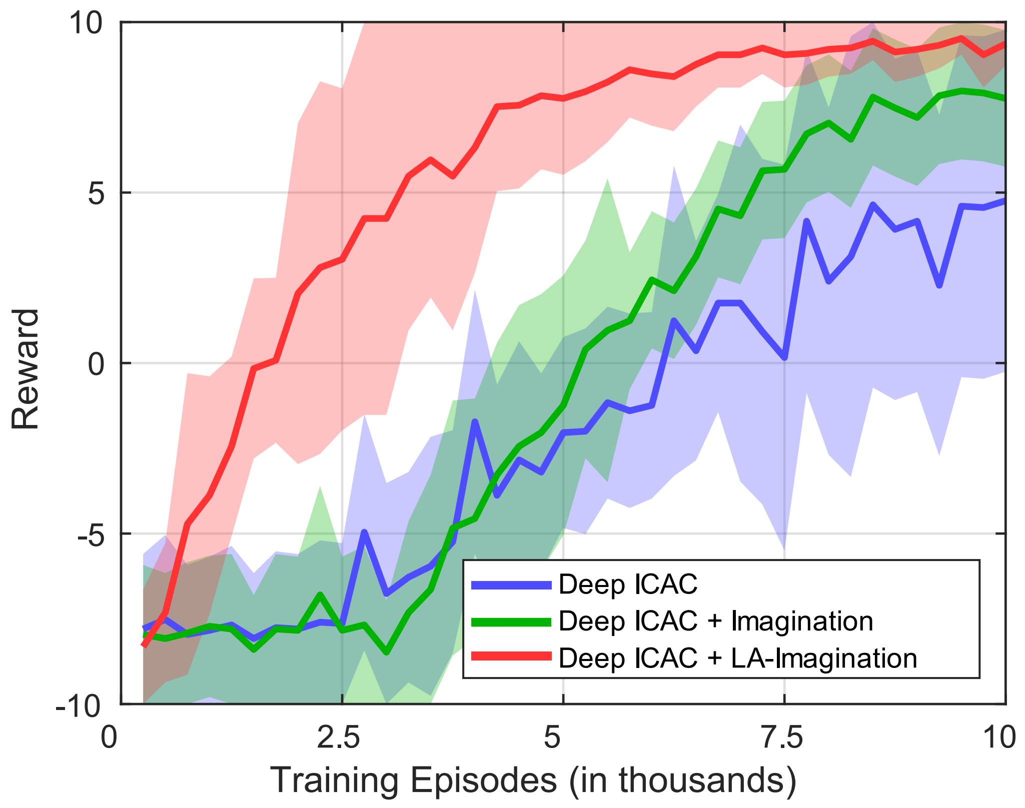


Figure 5.4: Learning curves of Deep ICAC without imagination, with static imagination, and with the proposed learning-adaptive imagination on robotic grasp learning from pixels. The curves are smoothed by averaging over a moving window of 250 episodes. Shaded regions correspond to one standard deviation.

Table 5.1: Learning speed (avg. reward per episode over the entire learning process) and convergence (avg. reward per episode over last 100 episodes).

	Deep ICAC	Deep ICAC + Imagination	Deep ICAC + LA-Imagination
Learning speed	-2.039	-0.548	**5.571**
Convergence	5.4	8.3	**9.4**

resulted in poor performance, even worse than the Deep ICAC baseline, over half of the learning process, as shown in the figure. In contrast, using learning-adaptive imagination led to significantly better performance, reaching higher rewards early and converging to a near-optimal policy in less than 6K episodes. In Table 5.1, we compare the learning speed and convergence (final performance) of the algorithms.

We also evaluate the effect of using different values of maximum imagination depth D^{max} on the learning performance. Figure 5.5 shows the mean episode extrinsic reward of Deep ICAC+LA-Imagination on the visual grasping task for different maximums of

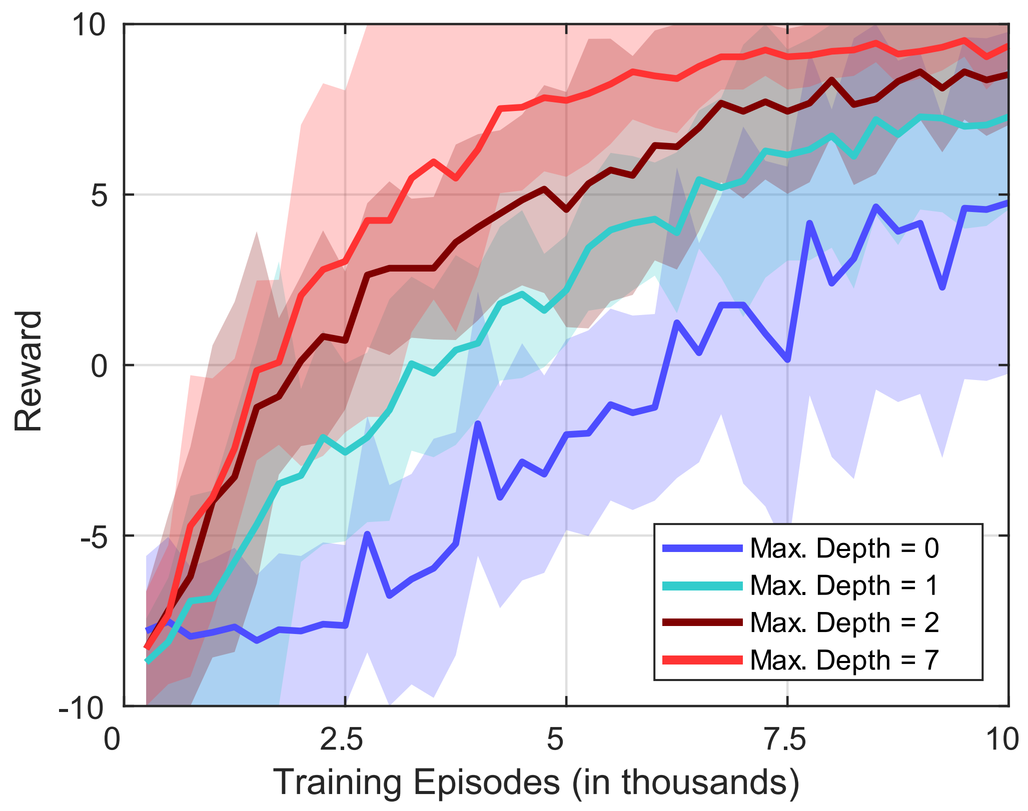


Figure 5.5: Learning curves of Deep ICAC+LA-Imagination for different values of maximum imagination depth.

imagination depth averaged over five runs. A rollout of a single imagined step was enough to improve the performance over the baseline (no imagination). Similarly, going from a maximum of one to two imagined steps allowed faster learning in the early episodes and led to a better final policy. Seven outperformed two, reaching higher reward after just 2K episodes. Values greater than seven did not change the performance. This is because the length of the imagined rollout is often shorter than a large D^{max}, as it stops increasing when the model prediction error is high before reaching D^{max}.

5.5 Conclusion

This chapter establishes a bridge between intrinsic motivation and imagination in robot decision-making, inspired by human mental simulation of motor behavior. Our approach performs imagination in a high-level latent space, resembling human imagination operating on abstract representations, to provide additional training experiences and accelerate skill learning. Unlike previous works, our approach generates imagined experiences only when the learned dynamics model and reward function have high local prediction accuracy, thus adapting to the learned underlying dynamics. In our approach, the imagination depth is adaptively determined using spatially and temporally local information provided by the average prediction error computed in different regions of the latent space over a recent time interval. We showed that integrating our approach to imagination with learning progress-based intrinsic motivation improves the sample efficiency of learning pixel-level control policies and achieves better final performance than the no-imagination and static-imagination baselines, particularly for robotic grasping in sparse reward environment.

Chapter 6

Integrated Imagination-Arbitration in Self-Organized Latent Space

6.1 Introduction

Dual-system approaches to robot motor learning have demonstrated how the advantages of model-based and model-free RL can be combined towards improving the sample efficiency of learning to perform complex robotic tasks. This line of approaches is also supported by convergent neural and behavioral findings on how the human brain arbitrates between model-based and model-free learning systems (Haith and Krakauer, 2013; Lee et al., 2014; Domenech and Koechlin, 2015). One common limitation to current dual-system robot learning approaches, however, is that they do not consider the prediction errors of the learned world model, which quickly accumulate when planning with the model and result in poor task performance.

In Chapter 4, we presented the Curious Meta-Controller (CMC), a meta-control algorithm that, unlike previous approaches, takes into account the reliability of model predictions when arbitrating online between model-based and model-free control systems. The reliability is measured by the model learning progress, which is the time derivative of the average prediction error of the model. If the reliability is positive, the meta controller queries the model-based system for an action, which performs gradient-based model predictive control, and if not, the model-free system is queried instead. While the results show that CMC can efficiently learn control policies from raw images, the approach relies only on the temporal information when computing the learning progress. It also uses a fixed planning horizon. Furthermore, the time and space complexity of gradient-based planning that CMC and other dual-system approaches use to infer an optimal action sequence under the model is very high. This is due to performing backpropagation through time to optimize the planning objective every time the model-based controller is queried for an action. Amos et al. (2018) attempt to address this issue by implicit differentiation of the Karush-Kuhn-Tucker optimality conditions at a fixed point of the convex optimization solver.

In this chapter, we propose a novel robot dual-system motor learning approach. Our approach improves on the previously proposed CMC algorithm by enabling an adaptive-length model rollout for plan optimization during model-based control through incrementally self-organizing the space of latent state representations and computing the reliability estimate locally for every region of the learned latent space. Rolling out the model until the estimated reliability is low, as opposed to using a fixed time horizon, ensures that no imperfect model predictions are used in computing the optimal plan and reduces the computational cost associated with gradient-based planning. The reliability estimate is used in computing an intrinsic reward to encourage actions that lead to data that improves the model. We also propose a unified learning framework that integrates online arbitration with offline experience imagination using the same underlying self-organized latent space, where imagined experiences collected from model rollouts are used as additional training data for the control policy. We evaluate our approach against baseline and state-of-the-art methods on learning vision-based robotic grasping in simulation and in the real world. The results show that our approach outperforms the compared methods and learns near-optimal grasping policies in dense and sparse reward environments.

6.2 Intrinsically Motivated Meta-Controller

Our dual-system motor learning approach consists of model-free and model-based control systems and a meta-controller deciding on which of the two systems to query for an action at each time step. We use the same model-free and model-based control systems presented in Section 4.4 (see Figure 4.1 and Figure 4.2). The model-free system is represented by an off-policy actor-critic learner. The jointly trained latent representation at the bottleneck layer of the convolutional autoencoder (Figure 4.1(a)) is used as input to the actor network (Figure 4.1(b)), which is trained according to the chosen off-policy actor-critic method; DDPG (Equation 4.2) or off-policy CACLA (Equation 4.3). The model-based system is represented by model predictive control (Figure 4.2) which performs gradient-based action optimization under a learned world model in latent space (Equation 4.6). Our approach to arbitrating between model-free and model-based control systems is based on the spatially and temporally local reliability in model predictions. We define the reliability in model predictions according to the average prediction error of the model in latent space. To improve model predictions, we use the change in the average prediction error as an intrinsic reward.

6.2.1 Local Reliability Estimation

We incrementally self-organize the latent space into local regions with local world models using the Instantaneous Topological Map (ITM) (Jockusch and Ritter, 1999) during motor exploration. ITM was originally designed for strongly correlated stimuli, which is the case here where the stimuli are the latent states visited along continuous trajectories, and has only a few hyperparameters. However, any other growing self-organizing network may also be used in our approach. The ITM network is defined by a set of nodes i, each having a weight vector w_i, and a set of edges connecting each node i to its neighbors $N(i)$. The network starts with two connected nodes, and when a new stimulus ϕ_s is observed, the following adaptation steps are performed:

1. Matching: Find the nearest node n and the second-nearest node n' to ϕ_s: $n \leftarrow \arg\min_{i} \left\| \phi_s - w_i \right\|_2^2$, $n' \leftarrow \arg\min_{j, j \neq n} \left\| \phi_s - w_j \right\|_2^2$.
2. Edge adaptation: If n and n' are not connected, add an edge between them. For all nodes $m \in N(n)$, if n' lies inside the Thales sphere through m and n (the sphere with diameter $w_m w_n$), $i.e. (w_n - w_{n'}) \cdot (w_m - w_{n'}) < 0$, remove the edge between m and n, and if m has no remaining edges, remove m.
3. Node adaptation: If ϕ_s lies outside the Thales sphere through n and n', $i.e. \left(w_n - \phi_s\right) \cdot \left(w_{n'} - \phi_s\right) > 0$, and if $\left\| \phi_s - w_n \right\|_2^2 > e_{max}$, where e_{max} is the desired mapping resolution, create a new node v with $w_v = \phi_s$ and an edge with n.

A moving window average of model prediction error is computed and updated separately for each latent-space region n (node in ITM):

$$\langle e_{t,n}^{prd} \rangle = \frac{1}{\sigma} \sum_{i=1}^{\sigma} e_i^{prd} \,\Big|_{e_i^{prd} = \left\| \mathcal{M}_n\left(\phi_{s_i}, a_i | \theta^{\mathcal{M}}\right) - \phi_{s_{i+1}} \right\|_2^2 + \left\| \mathcal{R}_n\left(\phi_{s_i}, a_i | \theta^{\mathcal{R}}\right) - r_i \right\|_2^2}, \tag{6.1}$$

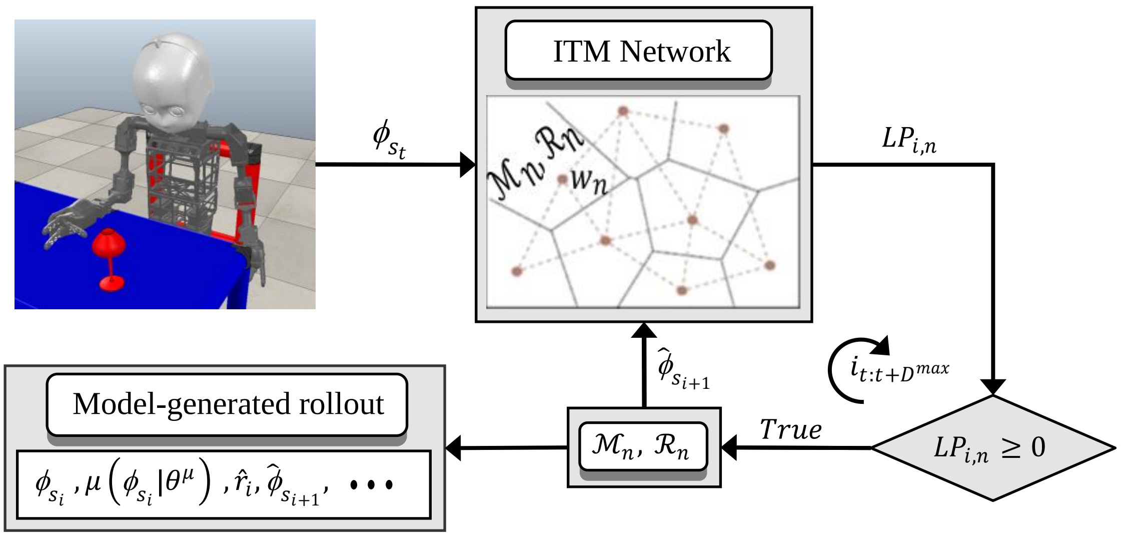
where σ specifies the length of the window of recent predictions in n, and $\mathcal{M}_n$ and $\mathcal{R}_n$ are the model's neural networks associated with n for predicting the next latent state and extrinsic reward respectively. The improvement in model predictions, the change in $\langle e_{t,n}^{prd} \rangle$ over time, is then estimated by computing the learning progress (LP) locally in each region using a time window $\mathcal{W}$:

$$LP_{t,n} = \langle e_{t-\mathcal{W},n}^{prd} \rangle - \langle e_{t,n}^{prd} \rangle. \tag{6.2}$$

The learning progress is used to derive an intrinsic reward $r_t^{int} = -LP_{t,n}$, encouraging actions that yield data that improves the model. It is also used as an unbiased, spatially and temporally local reliability estimator that underlies meta-decisions, as detailed in the following section.

6.2.2 Reliability-based Arbitration

When a new latent state ϕ_{s_t} is observed, the ITM network is updated and the nearest node n to ϕ_{s_t} is identified. If the corresponding learning progress in the latent region covered by n is negative, which indicates low prediction reliability for the local model, the meta-controller queries the model-free control system for an action. The model-free system in turn sends the output of the actor network $\mu\left(\phi_{s_t} | \theta^{\mu}\right)$ with exploration noise to the environment. If, on the other hand, the learning progress is greater or equal to zero, the meta-controller queries the model-based control system instead for an action. This initiates the plan optimization process (Figure 4.2). However, rather than using a predetermined planning horizon, the learning progress defined over the traversed latent regions the model-generated states belong to adaptively sets the depth of planning, as illustrated in Figure 6.1. This is done by terminating the model-generated rollout when the local learning progress is negative or a maximum depth is reached (see Algorithm 6). Rolling out the model until the estimated reliability is low ensures that no imperfect model predictions are used in computing the optimal plan and reduces the computational cost. The first action of the optimal plan is then sent to the environment with exploration noise. In either case and after performing an action a_t, the newly collected experience tuple (s_t, a_t, r_t, s_{t+1}),



ITM Network
ϕ_{s_t}
$\mathcal{M}_n$ $\mathcal{R}_n$ w_n
$LP_{i,n}$
$\hat{\phi}_{s_{i+1}}$
$i_{t:t+D^{max}}$
Model-generated rollout
$\phi_{s_i}, \mu(\phi_{s_i}|\theta^{\mu}), \hat{r}_i, \hat{\phi}_{s_{i+1}}, \dots$
$\mathcal{M}_n, \mathcal{R}_n$
True
$LP_{i,n} \geq 0$

Figure 6.1: Adaptive-length model rollout for model-based control: Given an initial latent state, ϕ_{s_t}, the world is unrolled using local models $\mathcal{M}_n$, where n is the nearest node to the initial real (and, later, model-generated) latent states, until the local prediction reliability estimated by the learning progress associated with the current latent region n is low or a maximum depth D^{max} is reached. At each rollout step i, the nearest node n to the model-generated latent state $\hat{\phi}_{s_{i+1}}$ is identified and the corresponding $LP_{i,n}$ determines whether to complete ($LP_{i,n} \geq 0$) or terminate ($LP_{i,n} < 0$) the rollout. The actions chosen in the rollout are the output of the actor network $\mu(.|\theta^{\mu})$. The predicted latent states $\hat{\phi}_{s_{i+1}}$ and rewards $\hat{r}_i$ are the outputs of the model's networks $\mathcal{M}$ and $\mathcal{R}$ respectively (Section 4.4.2). When the rollout is terminated, plan optimization is performed over the computed horizon ($H = i - t$), as illustrated in Figure 4.2.

Algorithm 6 Planning Depth (ϕ_{s_t}, n, D^{max})

```
1: $i \leftarrow 0,\ \phi_{s_i} \leftarrow \phi_{s_t}, LP_{i,n} \leftarrow LP_{t,n}$
2: while ($LP_{i,n} \geq 0$) and ($i \leq D^{max}$) do
3:     $\hat{a}_i \leftarrow \mu(\phi_{s_i}|\theta^{\mu}),\ \hat{\phi}_{s_{i+1}} \leftarrow \mathcal{M}_n(\phi_{s_i}, \hat{a}_i|\theta^{\mathcal{M}_n})$
4:     $\phi_{s_i} \leftarrow \hat{\phi}_{s_{i+1}}$
5:     $n \leftarrow$ best-matching node to $\phi_{s_i}$
6:     $i \leftarrow i + 1$
7: end while
8: return $i$
```

where $r_t = r_t^{ext} + r_t^{int}$, is added to the replay memory of recent experiences used to update the actor, critic-autoencoder, and world model networks. Figure 6.2 illustrates the arbitration process of the intrinsically motivated meta-controller.

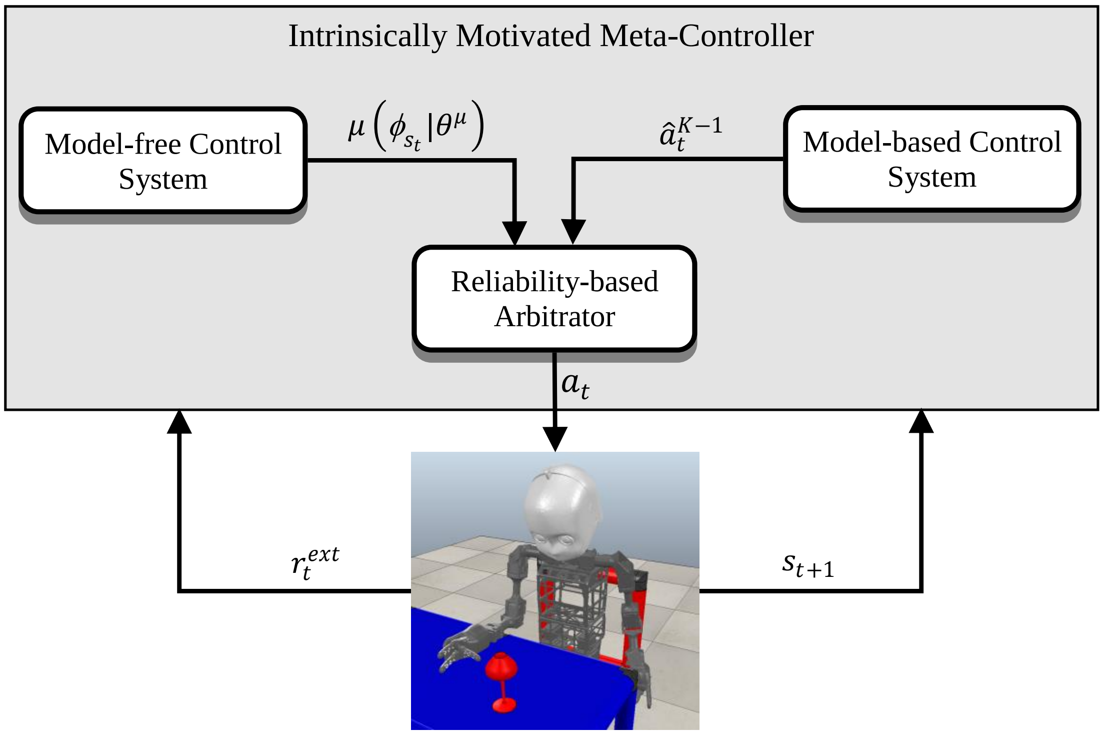

Intrinsically Motivated Meta-Controller
Model-free Control System
$\mu\left(\phi_{s_t}|\theta^{\mu}\right)$
$\hat{a}_t^{K-1}$
Model-based Control System
Reliability-based Arbitrator
a_t
r_t^{ext}
s_{t+1}

Figure 6.2: Intrinsically motivated meta-controller: At each time step, the learning progress in the latent-space region the current latent state ϕ_{s_t} belongs to is checked. If greater or equal to zero, this indicates high reliability in model predictions and the meta-controller queries the model-based control system for an action, which in turn performs plan optimization and returns the first action of the optimal plan $\hat{a}_t^{K-1}$. Otherwise, a negative learning progress indicates low prediction reliability and the meta-controller queries the model-free system for an action, which returns the output of the actor network $\mu\left(\phi_{s_t}|\theta^{\mu}\right)$. The selected action is then sent to the environment with exploration noise, and the environment returns the next state s_{t+1} and extrinsic reward r_t^{ext}.

In our approach, the model-free control system provides the model-based control system with a good initial action sequence. Likewise, the model-based control system provides the model-free control system with a better-informed exploratory action when the model is locally reliable. Thus, the two control systems are mutually beneficial. The complete algorithm for learning visuomotor control policies with our intrinsically motivated meta-controller is given in Algorithm 7.

6.3 Integrated Imagination-Arbitration Learning Framework

Besides planning, the predictive world models can be leveraged by generating imagined experience samples to augment real-world samples and improve the data efficiency of

Algorithm 7 Intrinsically Motivated Meta-Controller (IM2C)

```
 1: Input: max. planning depth $D^{max}$, no. of plan optimization iterations $K$, desired
           mapping resolution $e_{max}$, episode length $T$, no. of episodes $E$
 2: Given: an off-policy actor-critic method $\mathbb{AC}$
 3: Initialize the learning parameters $\{\omega, \tilde{\omega}, \theta^{Q}, \theta^{\mu}, \omega', \theta^{Q'}, \theta^{\mu'}\}$
 4: Initialize the ITM network with two connected nodes, $n1$ and $n2$, and the corre-
    sponding model parameters $\{\theta^{\mathcal{M}_{n1}}, \theta^{\mathcal{R}_{n1}}, \theta^{\mathcal{M}_{n2}}, \theta^{\mathcal{R}_{n2}}\}$
 5: Initialize replay buffer $B$
 6: for $episode = 1\ to\ E$ do
 7:     Sample initial state $s_1$
 8:     for $t = 1\ to\ T$ do
 9:       Compute latent state encoding $\phi_{s_t} = f(s_t \mid \omega)$
10:       Update the ITM network
11:       Identify best-matching node $n$
12:       if $LP_{t,n} \geq 0$ then
13:           $H \leftarrow$ Planning Depth $(\phi_{s_t}, n, D^{max})$ (see Algorithm 6)
14:           Query model-based control system with time horizon $H$ (see Section 4.4.2)
15:           $a_t \leftarrow \hat{a}_t^{K-1}$: $\hat{a}_t^{K-1}$ is the optimal plan's first action
16:       else
17:           Query model-free control system (see Section 4.4.1)
18:           $a_t \leftarrow \mu\left(\phi_{s_t} \mid \theta^{\mu}\right)$, where $\mu$ is $\mathbb{AC}$'s actor
19:       end if
20:       Add exploration noise $a_t \leftarrow a_t + \mathcal{N}(0, 1)$
21:       Execute $a_t$ and observe $r_t^{ext}$ and $s_{t+1}$
22:       Update $LP_{t,n}$ using Equation 6.2, and compute intrinsic reward $r_t^{int} = -LP_{t,n}$
23:       $r_t \leftarrow r_t^{ext} + r_t^{int}$
24:       Store $(s_t, \phi_{s_t}, a_t, r_t, r_t^{ext}, s_{t+1}, \phi_{s_{t+1}})$ in $B$
25:       Update $\{\theta^{\mathcal{M}_n}, \theta^{\mathcal{R}_n}\}$ using $(\phi_{s_t}, a_t, r_t^{ext}, \phi_{s_{t+1}})$ to minimize $\mathcal{L}_{model}$ (Equation 4.4)
26:       Update $\{\omega, \tilde{\omega}, \theta^{Q}\}$ on minibatch from $B$ to minimize $\mathcal{L}_{combined}$ (Equation 4.1)
27:       Update $\theta^{\mu}$ on minibatch from $B$ (Equation 4.2 or 4.3, depending on $\mathbb{AC}$)
28:       Update target network parameters: $\theta^{Q'} \leftarrow \tau\,\theta^{Q} + (1-\tau)\theta^{Q'}$,
          $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$, $\theta^{\mu'} \leftarrow \tau\,\theta^{\mu} + (1-\tau)\theta^{\mu'}$, with $\tau \ll 1$
29:     end for
30: end for
```

learning control policies. In Chapter 5, we demonstrated that performing imagined rollouts in a learned latent space and adapting the imagination depth to the improvement in learning a world model accelerates robotic visuomotor skill learning. Here, we propose to integrate our learning-adaptive imagination (LA-Imagination) with the presented reliability-based arbitration using the same underlying self-organized latent space.

In LA-Imagination (Section 5.3), an on-policy imagined rollout is performed every time step with a probability proportional to the local model's prediction accuracy. We modify the algorithm and instead use the adaptive-length model rollout (Figure 6.1), which is the input to plan optimization in our model-based control system, to provide a set of imagined transitions. To allow for learning from imagined latent-space transitions, we split the replay memory into pixel-space and latent-space replay buffers B_{pixel} and B_{latent} respectively. Real-world pixel-space transitions T_i^{pixel} are stored in B_{pixel} and used to learn the jointly optimized latent representation, while imagined latent-space transitions T_i^{latent} are stored in B_{latent} and used to learn the policy and action-value functions and the local world models. This is performed by updating parameters $\{\omega, \widetilde{\omega}\}$ with gradient descent on a minibatch from B_{pixel} to minimize $\mathcal{L}_{combined}$ (Equation 4.1) followed by updating parameters $\{\theta^Q\}$ with gradient descent on a minibatch from B_{latent} to minimize $\mathcal{L}_Q$, taking ϕ_{s_i} as an input, and updating parameters $\{\theta^\mu\}$ by following Equation 4.2 or 4.3 according to the chosen actor-critic method. In our proposed framework, offline learning from imagined transitions with experience replay is coupled with online meta control (discussed in Section 6.2) based on the spatially and temporally local model reliability estimated by the learning progress. Figure 6.3 shows the overall learning framework.

6.4 Experiments

In Sections 6.2 and 6.3, we have described the Intrinsically Motivated Meta-Controller (IM2C) and the Integrated Imagination-Arbitration (I2A) framework for improving data efficiency of learning robotic vision-based control policies. Here, we will evaluate their performance compared to baseline and state-of-the-art methods on robot grasp learning in simulation as well as on a real robot.

6.4.1 Evaluation in Simulation

Here, we describe the experimental setup, including the learning parameters and robotic environment, and the results of applying our proposed approaches and the compared algorithms to the simulated robot grasp-learning task.

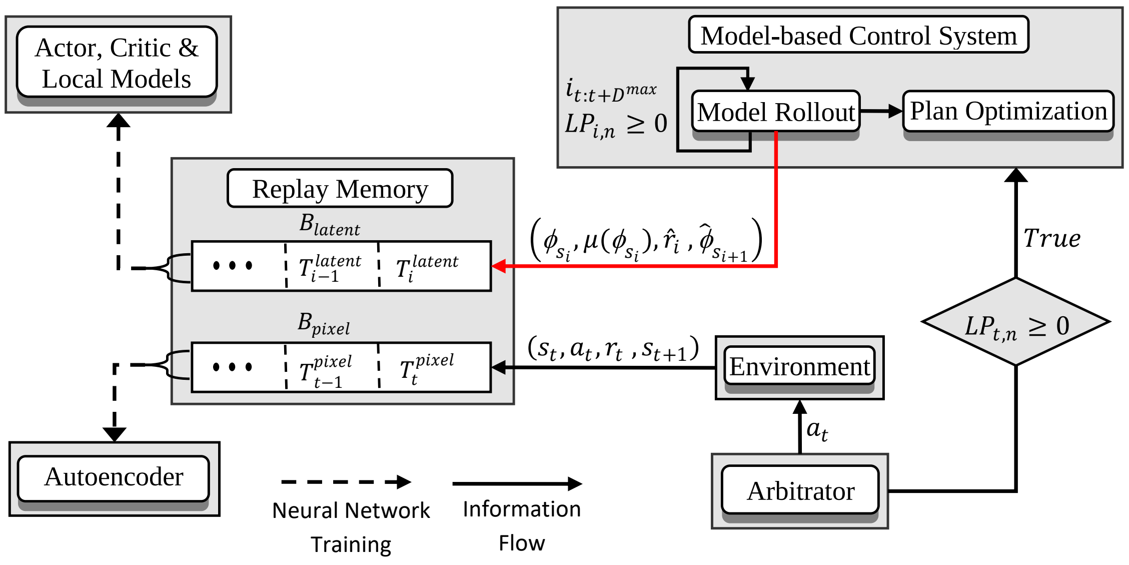


Figure 6.3: Integrated Imagination-Arbitration (I2A) framework: At each time step t, the Intrinsically Motivated Meta-Controller uses the learning progress $LP_{t,n}$ associated with node n to arbitrate between model-based and model-free control systems (Figure 6.2). If $LP_{t,n}$ is found to be greater or equal to zero, the model-based system is called and the model is unrolled in latent space until $LP_{i,n}$ is negative or a maximum depth D^{max} is reached. The resulting model rollout is used to provide a sequence of imagined transitions$\left(\phi_{s_i}, \mu(\phi_{s_i}), \hat{r}_i, \hat{\phi}_{s_{i+1}}\right)$, as shown by the red arrow, which are then added to the latent-space buffer B_{latent} and used in training the actor, critic and local model networks. It is also used as input to the plan optimization process of the model-based system. After arbitration, the action of the chosen control system is sent to the environment and the collected real-world transition $\{s_t, a_t, r_t, s_{t+1}\}$ is stored in B_{pixel} and used in training the autoencoder network to jointly optimize the reconstruction and value prediction losses.

Parameter and Implementation details.

We use the neural architectures shown in Figure 4.1 with the number and size of convolutional filters placed above the corresponding layers for representing the actor and critic in the considered algorithms. No pooling layers are used. All convolutional layers are zero-padded and have stride 1. ReLU activations are used in all layers except for the output layers of the actor and critic networks that use tanh and linear activations respectively. For representing the world model, we use a fully connected neural network with one hidden layer of 20 tanh units and two output layers of 32 and 1 linear units for predicting the next latent state and extrinsic reward respectively. The weighting coefficients λ_{rec} and λ_Q of the combined loss function defined in Equation 4.1 are set to 0.1 and 1 respectively. We set the learning rate α_{plan} (Equation 4.6), the number of gradient descent steps K and the maximum depth D^{max} of the plan optimization of the model-based control system to 1e-3, 10, and 6 respectively. A single replay buffer with a capacity of 100K transitions is used in all experiments except for the experiment with our proposed I2A method where we use two replay buffers B_{pixel} and B_{latent} with capacities of 60K and 200K respectively. All networks are trained from scratch using batch size 256 and Adam optimizer (Kingma and Ba, 2014) with learning rate 1e-3 for the critic-autoencoder and model networks and 1e-4 for the actor network. The discount factor γ and the update rate of the target networks τ are set to 0.99 and 1e-6 respectively. The desired mapping resolution e_{max} is set to 6 and the time windows used in computing the learning progress σ and $\mathcal{W}$ are set to 40 and 20 time units respectively. We train the networks using Tensorflow (Abadi et al., 2016) on a desktop with Intel i5-6500 CPU, 16 GB of RAM, and a single NVIDIA Geforce GTX 1050 Ti GPU.

Simulation environment.

All experiments are conducted on our Neuro-Inspired COmpanion (NICO) robot (Kerzel et al., 2017) using the V-REP robot simulator (Rohmer et al., 2013). NICO is a child-sized humanoid developed by the Knowledge Technology group of the University of Hamburg. NICO is a flexible platform for research on embodied neurocognitive models based on human-like sensory and motor capabilities. It stands about one meter tall; its body proportions and degrees of freedom resemble that of a three- to four-year-old child. Figure 6.4(a) shows the configuration of the environment, including the simulated NICO robot sitting in front of a table on top of which a glass is placed and used as the grasping target. In order to prevent self-collisions while still allowing for a large workspace, we consider learning a grasping policy that controls the shoulder joint and finger joints of the right hand, as shown in Figure 6.4(c). The shoulder joint has an angular range of movement of $\pm$ 100 degrees. The multi-fingered hand is tendon-operated and consists of 1 thumb and 2 index fingers with finger joints having an angular range of movement of $\pm$ 160 degrees. All algorithms take as input a 64×32 pixel RGB image obtained from the vision sensor whose output is shown in Figure 6.4(b).

Results.

We run the algorithms for 10K episodes. Each episode terminates when the target is grasped, toppled, or a maximum of 50 time steps is reached. The target position is randomly set to a new graspable position at the start of each episode.

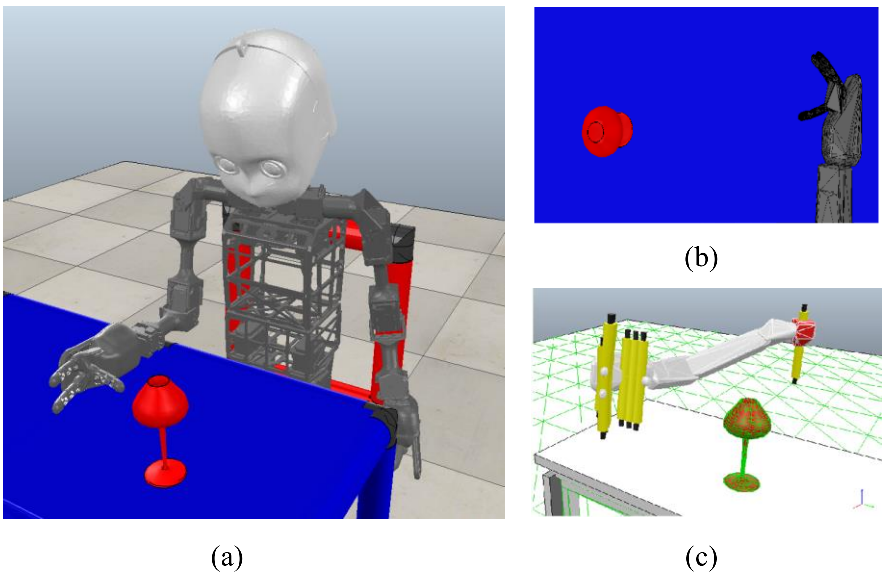
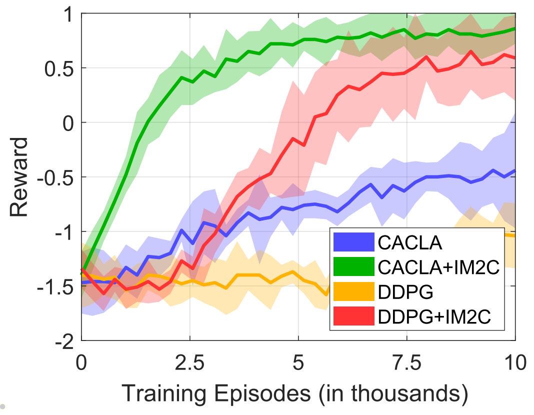


Figure 6.4: V-REP-simulated grasp-learning experiment: (a) NICO robot facing a table and attempting to grasp a glass randomly placed on the table, (b) the sensory input to the learning algorithm, and (c) the joints controlled by the grasping policy, depicted as yellow cylinders with one in the shoulder and 3 in each finger.

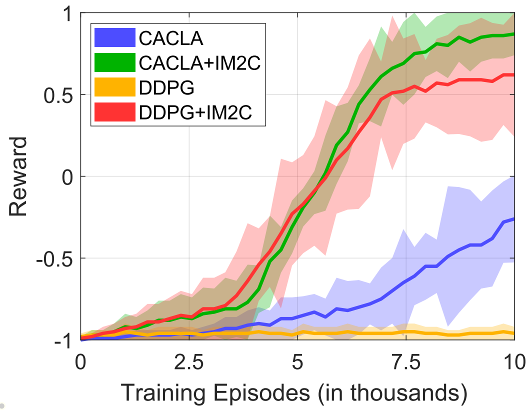
The extrinsic reward function is defined as follows:

$$r_t^{ext} = \begin{cases} +1 & \text{target grasped,} \\ -1 & \text{target toppled,} \\ -\|c^t - c^h\| & \text{otherwise (dense),} \\ 0 & \text{otherwise (sparse),} \end{cases}$$

where c^t and c^h are the center points of the target and the hand respectively. We compare the performance of off-policy CACLA and DDPG with and without our proposed IM2C on learning robotic vision-based grasping in dense and sparse reward settings. Figure 6.5 shows the episodic reward averaged over 5 random seeds. (We use the term episodic reward to refer to the sum of extrinsic rewards collected over one complete episode.) It can be observed that both CACLA+IM2C and DDPG+IM2C achieved a higher average episodic reward and a better convergence rate than their baseline counterparts at the end of training in both reward settings. The effect of IM2C is more evident in the results of learning from sparse rewards where CACLA+IM2C and DDPG+IM2C significantly outperformed their baseline counterparts in learning speed and final performance, as shown in Figure 6.5(b) and Table 6.1. We compute the following scoring metrics: (i) Area-under-Curve (AuC) is the area under the learning curve, normalized by the total area, and gives a quantitative measure of learning speed, and (ii) Final Performance (Final Perf) is the average episodic reward over the last 500 training episodes. The two metrics are reported in Table 6.1.



(a)



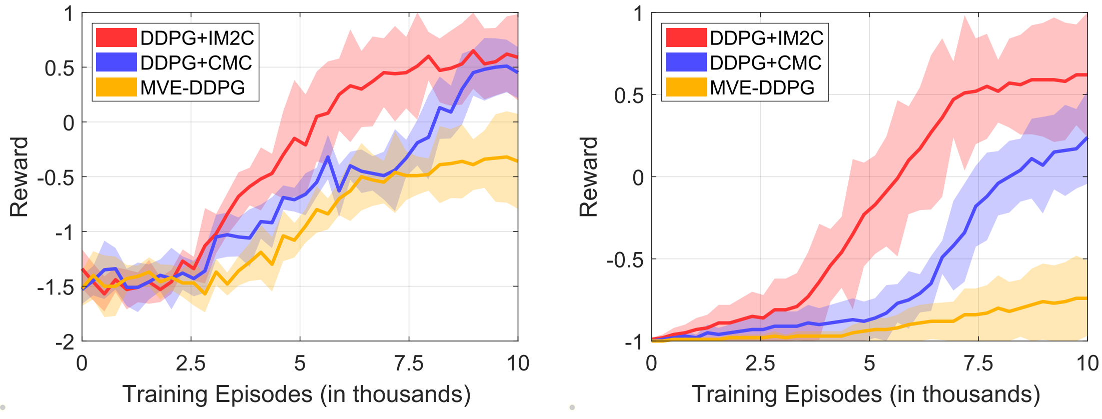
(b)

Figure 6.5: Learning curves of off-policy CACLA and DDPG with and without IM2C on robot grasp learning from pixel input in two reward settings: (a) dense reward and (b) sparse reward. The curves are smoothed using a sliding window of 250 episodes. Shaded regions correspond to one standard deviation.

Table 6.1: Summary statistics of the simulation results for different experimental settings.

	CACLA	CACLA+IM2C	DDPG	DDPG+IM2C
Dense Reward				
AuC	0.379	**0.815**	0.214	**0.554**
Final Perf.	-0.5±0.5	**0.8±0.1**	-1.0±0.3	**0.6±0.4**
Sparse Reward				
AuC	0.109	**0.440**	0.019	**0.406**
Final Perf.	-0.3±0.2	**0.9±0.1**	-0.9±0.0	**0.6±0.3**

We also compare IM2C to previous methods for improving model-free value estimation with model-based predictions, particularly the state-of-the-art Model-based Value Expansion (MVE) method (Feinberg et al., 2018) and the more recent Curious Meta-Controller (CMC) method (Hafez et al., 2019b), presented in Chapter 4. We implement MVE-DDPG from (Feinberg et al., 2018) and DDPG+CMC from (Hafez et al., 2019b) with a prediction horizon H of 2 and 3 steps respectively, and find these values to produce the best results. Figure 6.6 shows the average episodic reward over 5 random seeds. The three methods have a comparable learning performance over 3K episodes in the dense reward setting (Figure 6.6(a)). The episodic reward of DDPG+IM2C and DDPG+CMC, however, continues to increase faster than that of MVE-DDPG, reaching 0.59 and 0.45 respectively. In the sparse-reward setting (Figure 6.6(b)), MVE-DDPG shows no clear improvement in performance, while DDPG+IM2C and DDPG+CMC are able to improve

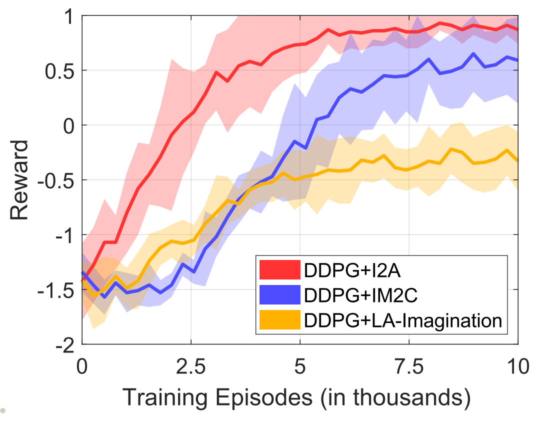


(a) (b)

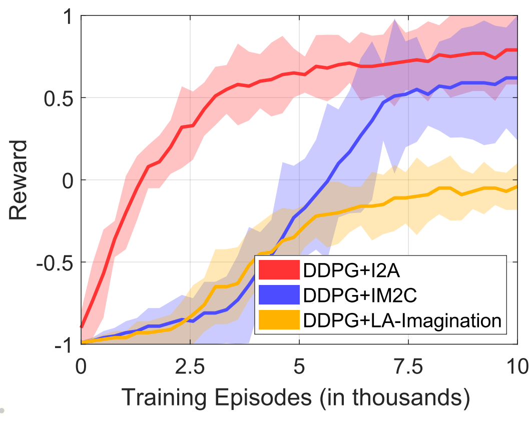
Figure 6.6: Learning curves of DDPG+IM2C, DDPG+CMC, and MVE-DDPG on robot grasp learning from pixel input in two reward settings: (a) dense reward and (b) sparse reward. The curves are smoothed using a sliding window of 250 episodes. Shaded regions correspond to one standard deviation.

their performance, converging to a policy of 0.62 and 0.24 episodic reward respectively. We believe the poor performance of MVE is primarily due to incorporating imperfect predictions in learning value estimates, as opposed to the reliability-driven model use of IM2C. Besides, CMC and MVE use fixed *H*, increasing the risk of compounding prediction errors, while IM2C enables automatic selection of *H* that is fully adaptive to the local reliability of the model.

Last but not least, we evaluate our proposed I2A framework, which combines experience imagination with reliability-based arbitration, by conducting an ablation study to analyze the influence of individual components of I2A, namely the arbitration and imagination components. This is performed by comparing I2A to IM2A that represents the arbitration component and to LA-Imagination (see Section 6.3) that represents the imagination component. The average episodic reward of running the three algorithms over 5 random seeds in the two reward settings is shown in Figure 6.7. It is clear that augmenting the replay memory with latent-space imagined transitions using LA-Imagination significantly improves the data efficiency of DDPG that completely failed to show any progress (see Figure 6.5). Compared to DDPG+LA-Imagination, DDPG+IM2C leads to a higher episodic reward, which again confirms the effectiveness of the meta controller in adaptively arbitrating between model-based and model-free control systems and choosing more informed exploratory actions, progressing faster to a good grasping policy. DDPG+I2A, on the other hand, yields the best results through combining the advantages of the two approaches using the same underlying self-organized latent space.



(a)



(b)

Figure 6.7: Learning curves of DDPG+I2A, DDPG+IM2C, and DDPG+LA-Imagination on robot grasp learning from pixel input in two reward settings: (a) dense reward and (b) sparse reward. The curves are smoothed using a sliding window of 250 episodes. Shaded regions correspond to one standard deviation.

6.4.2 Evaluation on a Real Robot

For the experiments with the physical NICO, the simulation environment was recreated as faithfully as possible: The simulation is based on a URDF model of NICO. Therefore, there is no difference in the simulated and the real robot. Both the table and NICO's seat have the same height as in the simulation, allowing for a direct transfer of the arm pose and, more importantly, the trained neural model. Furthermore, the color of the table is identical to the color in the simulation. A grasping object is slightly different in geometry, to allow for more stable grasps, but has the same color. To achieve the same perspective for the visual input, an external camera was mounted on the table with a view similar to the simulated camera. Figure 6.8 shows NICO in the experimental setup.

While the grasping object's position in the simulation environment can be manipulated directly and the virtual NICO is only used for grasp learning and execution, in the real environment, NICO is also used to place the object at an exact and known position on the table. Each grasping trial consists of the following steps: Starting from the initial position (shoulder at zero degrees), the grasping object is put into NICO's hand (if it was not already in the hand), the hand closes and NICO puts the object at a predetermined position on the table, the position is memorized and NICO moves the hand back to the starting position. Now the actual grasping trial starts by taking an image with the external camera and feeding it into the actor network that outputs a motor command to NICO. After the movement is executed, either the object is grasped, or if the hand is too far

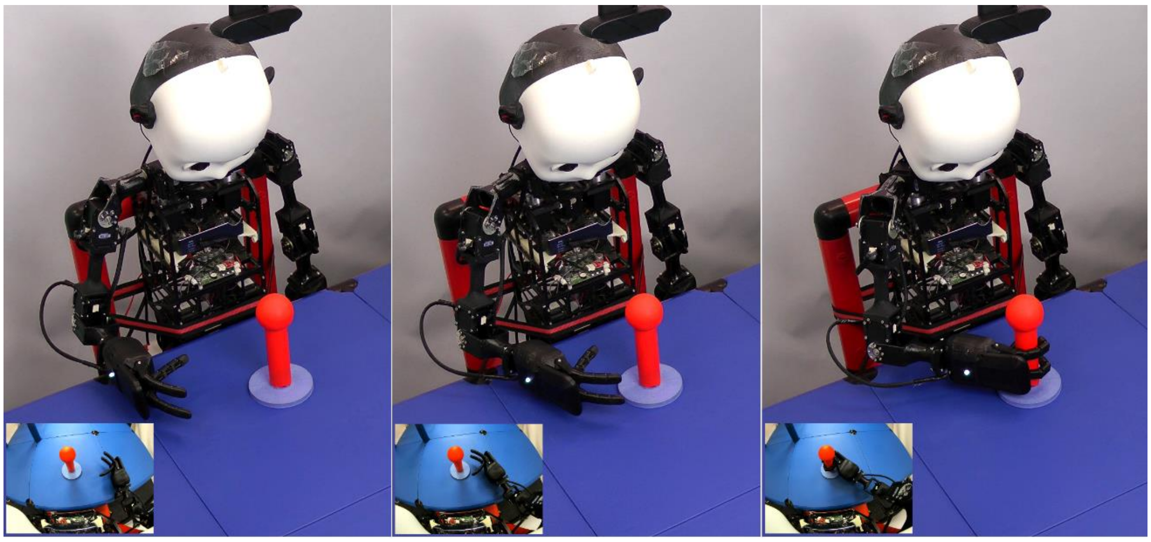


Figure 6.8: NICO experimental setup during a grasping test trial: From left to right: the exocentric and the egocentric (inset) views of the initial, intermediate, and full-grasp configurations.

from the object, another image is recorded, and the process is repeated. Up to eight consecutive grasping steps are performed before the attempt is categorized as failed, and the object is retrieved using its initially stored position.

To compare the performance of the algorithms on the real NICO robot, we take the best-performing policy network of each algorithm, trained in simulation, and then deploy it on the real robot. We perform 25 test episodes, each with a random graspable position. To achieve a seamless simulation-to-real transfer of the trained policy networks and to compensate for the slightly different alignment of the simulated and real cameras, we force the encoder part of the critic-autoencoder network to map one image from the simulation environment and one image from the real world with the same joint configuration and environmental setup into a similar latent representation. This is done by minimizing the Euclidean distance between latent representations, at the output of the encoder, corresponding to images from the simulated and real-world environments with supervised learning over a training set of 2K simulated-real image pairs. The encoder then computes the latent state to be used as input to the policy network during testing. No fine-tuning of the trained policy networks is performed. We report the success rate (the proportion of the successful test episodes) for each algorithm in Table 6.2.

Table 6.2: Success rate of the trained policy networks on the real robot.

Environment	DDPG	DDPG+CMC	MVE-DDPG	DDPG+IM2C	DDPG+I2A
Dense reward	16%	68%	48%	80%	**88%**
Sparse reward	12%	44%	12%	**76%**	**76%**

6.5 Conclusion

We presented a novel robot dual-system motor learning approach that is behaviorally and neurally plausible, data-efficient, and competitive with the state of the art. Our approach adaptively arbitrates between model-based and model-free decisions based on the spatially and temporally local reliability of a learned world model. The reliability estimate computed locally for every region of a learned latent space is used to make the meta-decision as well as to enable an adaptive-length model rollout for plan optimization during model-based control. We derive an intrinsic reward using the reliability estimate to encourage collecting experience data that improves the model. To further improve the data efficiency, we leverage the reliable multi-step model predictions by combining arbitration with experience imagination where imagined experiences collected from model rollouts are used as additional training data for the control policy.

We show that our approach learns better vision-based control policies than baseline and state-of-the-art methods in dense and sparse reward environments. Policy networks trained in simulation with our approach are shown to perform well on the physical robot without fine-tuning of the policy parameters. Our results suggest that model reliability is essential for dual-system approaches involving online meta-decisions to determine which of the model-based and model-free systems to query for an action and for generating imagined experience data that includes less overall prediction error. Our approach can be used with any off-policy reinforcement learning algorithm, which we demonstrated with off-policy CACLA and DDPG.

Chapter 7

Conclusion

Developing mechanisms of intrinsic motivation to improve internal knowledge of the world dynamics and using them for learning complex robotic visuomotor skills is a challenging but important step forward on the path towards building truly autonomous robots. In this thesis, we proposed approaches that tackle some of the main issues in this area. Here, we particularly describe how each of these approaches addresses each of the research objectives of the thesis.

To address the objective of providing a directed exploration strategy, we proposed a spatially and temporally local learning progress defined as the time derivative of the average prediction error of a local dynamics model, as presented in Chapter 2. The learning progress computed locally in each self-organized sensory region is used to derive an intrinsic reward to encourage the RL agent to direct its exploration from regions of highly predictable sensorimotor dynamics to regions of less predictable dynamics, resulting in directed and data-efficient exploration strategy. The directed exploration is then integrated with a continuous actor-critic architecture and shown to lead to fast and stable learning of control policies, particularly in environments with sparse rewards.

To address the objective of reducing the sample complexity of learning vision-based control policies, we proposed in Chapter 3 two neural architectures for learning efficient, task-relevant state representations from high-dimensional observations. A hierarchical

SFA network is first proposed for unsupervised learning of low-dimensional state representations that encode important invariances in the raw observations. The presented SFA-based ICAC algorithm, which uses the SFA-trained state representations and the spatially and temporally local learning progress, is shown to achieve near-optimal performance on learning vision-based robotic reaching policies in a relatively small number of training episodes, reducing the sample complexity. Second, a novel jointly trained deep neural architecture for learning a low-dimensional state representation is proposed. This representation, which is trained to minimize the joint autoencoder's reconstruction and critic's value prediction loss, captures the information necessary to reconstruct the original input and recognize states that lead to high rewards and is therefore used as input to the actor network. Training the actor directly on the low-dimensional representation allows for more sample-efficient learning of the target policy. The local world models whose predictions are used for computing the learning progress-based intrinsic reward are trained in the space of the jointly optimized state representations. The resulting algorithm, which we call Deep ICAC, is shown to outperform the state-of-the-art and baseline algorithms on learning robot reaching and grasping skills from raw pixels in simulation and in the real world.

To address the objective of designing an unbiased model reliability estimate for adaptive arbitration between model-based and model-free control, we proposed in Chapter 4 to train a model of the world dynamics in a learned latent space and derive an intrinsic reward based on the learning progress of the model, representing the agent's curiosity to take actions that lead to data that improves the model. The learning progress is used as an unbiased reliability estimator that underlies the decision which of the model-free and model-based control systems to query for an action at each time step, resulting in the Curious Meta-Controller algorithm. Unlike previous works, our algorithm considers the reliability of the model when arbitrating between the two control systems. The experimental results show that using the curious meta-controller improves the efficiency of learning pixel-level robotic reaching and grasping policies.

To address the objective of reliably generating imagined experiences, we proposed in Chapter 5 a learning-adaptive imagination approach that performs experience imagination in a learned latent space. In our approach, the latent space is self-organized into local regions with local dynamics models, and a running average of model prediction error is maintained for each region. Imagined rollouts are generated under the current model with probability inversely proportional to the average error of the current region, and the imagination depth is adaptively determined by the average error of the traversed regions. To encourage collecting data that improves model predictions necessary for imagination, we use an intrinsic reward based on the spatially and temporally local learning progress. In order to use the latent-space imagined experiences in the training, the experience replay buffer is divided into pixel-space and latent-space buffers for storing real and imagined experiences respectively. As a result, the policy is trained on additional imagined data generated reliably by rolling out the learned models in regions of low average prediction errors, meeting our objective. The results show that our approach achieves better final performance than the no-imagination and static-imagination baselines, particularly for robotic grasping in sparse reward environments.

To address the last objective of developing a dual-system learning approach that enables an adaptive-length model rollout for plan optimization during model-based control, we proposed in Chapter 6 to improve upon the Curious Meta-Controller approach presented in Chapter 4 by incrementally self-organizing the space of latent state representations and computing the reliability estimate locally for every region of the self-organized latent space. Instead of using a fixed time horizon, rolling out the model until the estimated reliability is low ensures that imperfect model predictions are not used in computing the optimal plan and reduces the computational cost associated with gradient-based planning. An intrinsic reward is derived using the reliability estimate to encourage collecting experience data that improves the model. To further improve the data efficiency, we leverage the reliable multi-step model predictions by combining arbitration with experience imagination where imagined experiences collected from model rollouts are used as additional training data for the control policy. The results show that our dual-system approach learns better vision-based control policies than baseline and state-of-the-art methods in dense and sparse reward environments. Also, policy networks trained in simulation with our approach are shown to perform well on the physical robot without fine-tuning of the policy parameters.

While the approaches presented in this thesis have contributed towards reducing the sample complexity of learning motor skills purely from the raw visual input and with no pretrained control policies in simulation and the real world, they incur an increased computational cost. This cost is basically introduced by the process of updating the growing self-organizing neural network (the ITM network) each time a state transition is observed, which is necessary for the computation of our local learning progress. Particularly, this involves the matching step that scales with the number of nodes in the network and the edge adaptation step that scales with the average number of neighboring nodes. All other operations are independent of the number of nodes. This incurred cost is minimal when the average size of the self-organizing network is small, which can be controlled through the desired mapping resolution hyperparameter.

All predictive models we used in this thesis as the basis for computing the learning progress and for performing action planning are single-step models. We believe that our proposed approaches can be extended to the case of a multi-step model, instead of the single-step model, by incorporating temporal abstractions, such as options (Sutton et al., 1999; Precup, 2000). Similarly, the control policy and model neural networks used in our approaches do not have a hierarchical structure which is naturally present in human decision-making, particularly for planning high-level actions over long time horizons. Integrating such a structure into our approaches would allow for learning control at multiple time scales in hierarchically structured tasks. Another promising direction for future work is to generalize our approaches to environments with stochastic dynamics.

Last but not least, our results suggest that model reliability is essential for dual-system approaches involving online arbitration between model-based and model-free control systems. This indicates that our reliability-based arbitration approach could have an important role to play in other problem domains, such as controlling semi-autonomous vehicles that pose strict safety constraints and require a mediator to arbitrate between the human driver and the automated system.

Finally, the approaches presented in Chapters 5 and 6 build a bridge between intrinsic motivation and experience imagination in robot decision-making, which we have shown to significantly improve the sample efficiency of robotic visuomotor skill learning. We believe that this will stimulate future research to investigate the feasibility and potential of utilizing the approaches in other robot learning applications.

Appendix A

List of Abbreviations

A3C	Asynchronous Advantage Actor-Critic
AuC	Area-under-Curve
CACLA	Continuous Actor-Critic Learning Automaton
CMC	Curious Meta-Controller
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DoF	Degree of Freedom
DQN	Deep Q-Network
I2A	Integrated Imagination-Arbitration
ICAC	Intrinsically motivated Continuous Actor-Critic
IM2C	Intrinsically Motivated Meta-Controller
ITM	Instantaneous Topological Map
LP	Learning Progress
MDP	Markov Decision Process
MPC	Model Predictive Control

MVE	Model-based Value Expansion
NICO	Neuro-Inspired Companion
PER	Prioritized Experience Replay
RL	Reinforcement Learning
SFA	Slow Feature Analysis
SOM	Self-Organizing Map
SR	Successor Representation
TD	Temporal Difference
TDM	Temporal Difference Model

Appendix B

Supplementary Algorithms

Algorithm 8 CACLA (Van Hasselt, 2012)

```
 1: Initialize actor and critic parameters $\theta^{\mu}$ and $\theta^{V}$
 2: for $t \in \{0,1,2,\dots\}$ do
 3:     Choose $a_t \sim \pi$: $\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(a_t - \mu(s_t|\theta^{\mu})\right)^2/2\sigma^2}$
 4:     Perform $a_t$, observe $r_{t+1}$ and $s_{t+1}$
 5:     $\delta_t = r_t + \gamma V(s_{t+1}|\theta^{V}) - V(s_t|\theta^{V})$
 6:     $\theta^{V} \leftarrow \theta^{V} + \alpha\delta_t \nabla_{\theta^{V}} V(s_t|\theta^{V})$
 7:     if $\delta_t > 0$ then
 8:         $\theta^{\mu} \leftarrow \theta^{\mu} + \beta\left(a_t - \mu(s_t|\theta^{\mu})\right)\nabla_{\theta^{\mu}}\mu(s_t|\theta^{\mu})$
 9:     end if
10:     if $s_{t+1}$ is terminal then
11:         Reinitialize $s_{t+1}$
12:     end if
13: end for
```

Algorithm 9 DDPG (Lillicrap et al., 2016)

```
 1: Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$
 2: Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
 3: Initialize replay buffer $R$
 4: for episode = 1, $M$ do
 5:   Initialize a random process $\mathcal{N}$ for action exploration
 6:   Receive initial observation state $s_1$
 7:   for $t$ = 1, $T$ do
 8:     Select $a_t = \mu(s_t|\theta^\mu)$ according to the current policy and exploration noise
 9:     Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
10:     Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
11:     Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
12:     Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
13:     Update critic by minimizing the loss $L = \frac{1}{N}\sum_i \big(y_i - Q(s_i, a_i|\theta^Q)\big)^2$
14:     Update the actor policy using the sampled policy gradient:
            $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_i}$
15:     Update the target networks (with $\tau \ll 1$):
            $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
            $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16:   end for
17: end for
```

Algorithm 10(a) Proportional PER (Schaul et al., 2016) for DDPG

```
 1: **Input:** minibatch $k$, step-size $\eta$, replay period $K$ and size $N$, exponents $\alpha$ and $\beta$
 2: Initialize replay memory $R = \emptyset$, $\Delta_c = 0$, $\Delta_a = 0$, $p_1 = 1$
 3: Observe initial state $s_1$
 4: **for** $t = 1$ to $T$ **do**
 5:    Choose $a_t = \mu(s_t|\theta^\mu)$ and observe $r_t$ and $s_{t+1}$
 6:    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$ with maximal priority $p_t = \max_{i<t} p_i$
 7:    **if** $t \equiv 0$ (mod $K$) **then**
 8:       **for** $j = 1$ to $k$ **do**
 9:          Sample transition $j \sim P(j) = p_i^\alpha / \sum_k p_k^\alpha$
10:          Compute importance-sampling weight $w_j = 1/(N \cdot P(j))^\beta$
11:          Compute TD-error $\delta_j = r_j + \gamma Q'\big(s_{j+1}, \mu'(s_{j+1}|\theta^{\mu'})|\theta^{Q'}\big) - Q\big(s_j, a_j|\theta^Q\big)$
12:          Update transition priority $p_j \leftarrow |\delta_j|$
13:          Accumulate critic weight-change $\Delta_c \leftarrow \Delta_c + w_j.\delta_j.\nabla_{\theta^Q} Q\big(s_j, a_j|\theta^Q\big)$
14:          Accumulate actor weight-change
                $\Delta_a \leftarrow \Delta_a + w_j.\nabla_a Q(s, a|\theta^Q)|_{s=s_j, a=\mu(s_j)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_j}$
15:       **end for**
16:       Update weights $\theta^Q \leftarrow \theta^Q + \eta.\Delta_c$, $\theta^\mu \leftarrow \theta^\mu + \eta.\Delta_a$, reset $\Delta_c = \Delta_a = 0$
17:       Update target networks $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$, $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
18:    **end if**
19: **end for**
```

Algorithm 10(b) Proportional PER (Schaul et al., 2016) for CACLA

```
 1: **Input:** minibatch $k$, step-size $\eta$, replay period $K$ and size $N$, exponents $\alpha$ and $\beta$
 2: Initialize replay memory $R = \emptyset$, $\Delta_c = 0$, $\Delta_a = 0$, $p_1 = 1$
 3: Observe initial state $s_1$
 4: **for** $t = 1$ to $T$ **do**
 5:    Choose $a_t \sim \pi$: $\pi(a_t|s_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(a_t - \mu(s_t|\theta^\mu))^2/2\sigma^2}$ and observe $r_t$ and $s_{t+1}$
 6:    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$ with maximal priority $p_t = \max_{i<t} p_i$
 7:    **if** $t \equiv 0$ (mod $K$) **then**
 8:       **for** $j = 1$ to $k$ **do**
 9:          Sample transition $j \sim P(j) = p_i^\alpha / \sum_k p_k^\alpha$
10:          Compute importance-sampling weight $w_j = 1/(N \cdot P(j))^\beta$
11:          Compute TD-error $\delta_j = r_j + \gamma V\big(s_{j+1}|\theta^V\big) - V\big(s_j|\theta^V\big)$
12:          Update transition priority $p_j \leftarrow |\delta_j|$
13:          Accumulate critic weight-change $\Delta_c \leftarrow \Delta_c + w_j.\delta_j.\nabla_{\theta^V} V\big(s_{j+1}|\theta^V\big)$
14:          Accumulate actor weight-change $\Delta_a \leftarrow \Delta_a + w_j.\big(a_t - \mu(s_t|\theta^\mu)\big)\nabla_{\theta^\mu}\mu(s_t|\theta^\mu)$
15:       **end for**
16:       Update weights $\theta^V \leftarrow \theta^V + \eta.\Delta_c$, $\theta^\mu \leftarrow \theta^\mu + \eta.\Delta_a$, reset $\Delta_c = \Delta_a = 0$
17:    **end if**
18: **end for**
```

Algorithm 11 MVE-DDPG (Feinberg et al., 2018)

```
1: Initialize targets $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$
2: Initialize the replay buffer $\beta \leftarrow \emptyset$
3: **while** not tired **do**
4:    Collect transitions $\tau = (s, a, r, s')$ from any exploratory policy
5:    Add observed transitions to $\beta$
6:    Fit the dynamics $\hat{f} \leftarrow \underset{f}{\operatorname{argmin}} \underset{\beta}{\mathbb{E}}[\|f(s,a) - s'\|^2]$
7:    **for** a fixed number of iterations **do**
8:       Sample $\tau_0 \sim \beta$: $\tau_0 = (s_{-1}, a_{-1}, r_{-1}, s_0)$
9:       Update $\theta^{\mu}$ with $\nabla_{\theta^{\mu}} Q(s, \mu(s|\theta^{\mu})|\theta^{Q})|_{s=s_{-1}}$
10:      Generate future transitions under the approximate model $\hat{f}$ for $t \in [H-1]$
             $\tau_t = (s_{t-1}, a_{t-1}, r(s_{t-1}, a_{t-1}), \hat{f}(s_{t-1}, a_{t-1})): a_{t-1} = \mu(s_{t-1}|\theta^{\mu'})$
11:      Update $\theta^{Q}$ by gradient descent on
             $$\frac{1}{H}\sum_{t=-1}^{H-1}\left(Q(\hat{s}_t, \hat{a}_t|\theta^{Q}) - \left(\sum_{k=t}^{H-1}\gamma^{k-t}\hat{r}_k + \gamma^{H} Q(\hat{s}_H, \hat{a}_H|\theta^{Q'})\right)\right)^2,$$
             $where\ \hat{s}_t = \hat{f}(s_{t-1}, a_{t-1}), \hat{a}_t = \mu\left(\hat{s}_t|\theta^{\mu'}\right), \hat{r}_k = r(\hat{s}_k, \hat{a}_k)$
12:      Update targets $\theta^{Q'}$ and $\theta^{\mu'}$ with some decay
13:   **end for**
14: **end while**
```

Appendix C

Publications Originating from this Thesis

Journal Articles

- Hafez, M. B., Weber, C., Kerzel, M., Wermter, S. (2020). Improving Robot Dual-System Motor Learning with Intrinsically Motivated Meta-Control and Latent-Space Experience Imagination. Submitted to Robotics and Autonomous Systems—arXiv preprint arXiv:2004.08830.

- Hafez, M. B., Weber, C., Kerzel, M., Wermter, S. (2019). Deep intrinsically motivated continuous actor-critic for efficient robotic visuomotor skill learning. Paladyn, Journal of Behavioral Robotics, 10(1), 14–29.

Conference Papers

- Hafez, M. B., Weber, C., Kerzel, M., Wermter, S. (2019). Efficient Intrinsically Motivated Robotic Grasping with Learning-Adaptive Imagination in Latent Space. In Proceedings of the Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob), pages 240–246, Oslo, Norway.

- Hafez, M. B., Weber, C., Kerzel, M., Wermter, S. (2019). Curious Meta-Controller: Adaptive Alternation between Model-Based and Model-Free Control in Deep Reinforcement Learning. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), pages 1–8, Budapest, Hungary.

- Hafez, M. B., Kerzel, M., Weber, C., Wermter, S. (2018). Slowness-based neural visuomotor control with an Intrinsically motivated Continuous Actor-Critic. In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), pages 509–514, Bruges, Belgium.

- Hafez, M. B., Weber, C., Wermter, S. (2017). Curiosity-Driven Exploration Enhances Motor Skills of Continuous Actor-Critic Learner. In Proceedings of the Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob), pages 39–46, Lisbon, Portugal.

Appendix D

Acknowledgements

I would first and foremost like to thank my supervisor Prof. Dr. Stefan Wermter for his valuable advice and constant support throughout my doctoral studies. I feel very fortunate to have had the opportunity to learn from someone who combines insightful scientific vision and extraordinary breadth of knowledge with unbounded kindness. Additionally, I would like to thank Dr. Cornelius Weber and Dr. Matthias Kerzel for the helpful discussions and feedback.

I would also like to thank my fellow colleagues of the Knowledge Technology research group for creating a very friendly, stimulating, and pleasant environment, both within and outside the lab. A very special thanks to Katja Kösters and Erik Strahl for their skilled administrative and technical assistance respectively.

I am deeply grateful to the German Academic Exchange Service (DAAD) for their generous financial support of my doctoral studies under the funding programme Research Grants–Doctoral Programmes in Germany (No. 57214224). I also gratefully acknowledge the partial support from the German Research Foundation (DFG) under project CML (TRR 169).

Finally, I would like to thank my family, especially my mom and dad, for their unconditional love, understanding, and encouragement throughout my doctoral studies; I am forever grateful for their support.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., and Kudlur, M. (2016). Tensorflow: a system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283.

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J.Z. (2018). Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Systems (NIPS)*, Montreal, Canada, pages 8289–8300.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in Neural Information Systems (NIPS)*, Long Beach, CA, USA, pages 5048–5058.

Baranes, A. and Oudeyer, P.Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73.

Barron, T., Obst, O., and Amor, H.B. (2017). Information maximizing exploration with a latent dynamics model. In *Conference on Neural Information Processing Systems, Deep RL Symposium (NIPS)*, Long Beach, CA, United States.

Boureau, Y.L., Sokol-Hessner, P., and Daw, N.D. (2015). Deciding how to decide: Self-control and meta-decision making. *Trends in Cognitive Sciences*, 19(11):700–710.

Brafman, R.I. and Tennenholtz., M. (2002). R-MAX a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231.

Cangelosi, A. and Schlesinger, M. (2015). *Developmental robotics: From babies to robots*, Cambridge, MA: MIT Press.

Case, L.K., Pineda, J., and Ramachandran, V.S. (2015). Common coding and dynamic interactions between observed, imagined, and experienced motor and somatosensory activity. *Neuropsychologia*, 79:233–245.

Chentanez, N., Barto, A.G., and Singh, S. (2005). Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS),* Vancouver, BC, Canada, pages 1281–1288.

Cushman, F. and Morris, A. (2015). Habitual control of goal selection in humans. *Proceedings of the National Academy of Sciences (PNAS)*, 112(45):13817–13822.

Daw, N.D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711.

Domenech, P. and Koechlin, E. (2015). Executive control and decision-making in the prefrontal cortex. *Current Opinion in Behavioral Sciences*, 1:101–106.

Driskell, J.E., Copper, C., and Moran, A. (1994). Does mental practice enhance performance? *Journal of Applied Psychology*, 79(4):481–492.

Elman, J.L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Fard, F.S. and Trappenberg, T.P. (2018). Mixing habits and planning for multi-step target reaching using arbitrated predictive actor-critic. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN),* Rio de Janeiro, Brazil, pages 1–8.

Feinberg, V., Wan, A., Stoica, I., Jordan, M.I., Gonzalez, J.E., and Levine, S. (2018). Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*.

Finn, C., Tan, X.Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, pages 512–519.

Florensa, C., Duan, Y., and Abbeel, P. (2017). Stochastic neural networks for hierarchical reinforcement learning. In *proceedings of International Conference on Learning Representations (ICLR)*, Toulon, France.

Forestier, S., Mollard, Y., and Oudeyer, P.Y. (2017). Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*.

Forestier, S. and Oudeyer, P.Y. (2016). Modular active curiosity-driven discovery of tool use. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, pages 3965–3972.

François-Lavet, V., Bengio, Y., Precup, D., and Pineau, J. (2019). Combined reinforcement learning via abstract representations. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, United States, pages 3582–3589.

Franzius, M., Wilbert, N., and Wiskott, L. (2011). Invariant object recognition and pose estimation with slow feature analysis. *Neural Computation*, 23(9):2289–2323.

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, 7:625–632.

Fu, J., Co-Reyes, J., and Levine, S. (2017). Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in Neural Information Systems (NIPS)*, Long Beach, CA, USA, pages 2577–2587.

Garcia, F.M. and Thomas, P. (2019). A meta-MDP approach to exploration for lifelong reinforcement learning. In *the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Montreal, Canada, pages 1976-1978.

Gottlieb, J., Oudeyer, P.Y., Lopes, M., and Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in Cognitive Sciences*, 17:585–593.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep Q-learning with model-based acceleration. In *Proceedings of the International Conference on Machine Learning (ICML)*, New York, United States, pages 2829–2838.

Hafez, M.B. and Loo, C.K. (2015). Topological Q-learning with internally guided exploration for mobile robot navigation. *Neural Computing and Applications*, 26(8):1939–1954.

Hafez, M.B., Weber, C., Kerzel, M., and Wermter, S. (2019b). Curious Meta-Controller: Adaptive Alternation between Model-Based and Model-Free Control in Deep Reinforcement Learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, pages 1–8.

Hafez, M.B., Weber, C., Kerzel, M., and Wermter, S. (2019a). Deep intrinsically motivated continuous actor-critic for efficient robotic visuomotor skill learning. *Paladyn, Journal of Behavioral Robotics*, 10(1):14–29.

Haith, A.M. and Krakauer, J.W. (2013). Model-based and model-free mechanisms of human motor learning. In *Progress in Motor Control*, Springer, New York, NY, pages 1–21.

Hamrick, J.B. (2019). Analogues of mental simulation and imagination in deep learning. *Current Opinion in Behavioral Sciences*, 29:8–16.

Ha, D. and Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

Hazan, E., Kakade, S.M., Singh, K., and Van Soest, A. (2019). Provably efficient maximum entropy exploration. In *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, United States, pages 2681–2691.

Hester, T. and Stone, P. (2015). Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F.D., and Abbeel, P. (2016). VIME: variational information maximizing exploration. In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain, pages 1109–1117.

Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., and Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France.

Jockusch, J. and Ritter, H. (1999). An instantaneous topological mapping model for correlated stimuli. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Washington, pages 529–534.

Kalweit, G. and Boedecker, J. (2017). Uncertainty-driven imagination for continuous deep reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning, volume 78 of Proceedings of Machine Learning Research*, Mountain View, United States, pages 195–206.

Keramati, M., Smittenaar, P., Dolan, R.J., and Dayan, P. (2016). Adaptive integration of habits into depth-limited planning defines a habitual goal-directed spectrum. *Proceedings of the National Academy of Sciences (PNAS)*, 113(45):12868–12873.

Kerzel, M., Strahl, E., Magg, S., Navarro-Guerrero, N., Heinrich, S., and Wermter, S. (2017). 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)*, pages 113–120.

Kerzel, M. and Wermter, S. (2017). Neural end-to-end self-learning of Visuomotor skills by environment interaction. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 27–34.

Ke, N.R., Singh, A., Touati, A., Goyal, A., Bengio, Y., Parikh, D., and Batra, D. (2019). Learning dynamics model in reinforcement learning by incorporating the long term future. In *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, United States.

Kidd, C., Piantadosi, S.T., and Aslin, R.N. (2012). The Goldilocks effect: Human infants allocate attention to visual sequences that are neither too simple nor too complex. *PloS one*, 7(5):e36399.

Kidd, C., Piantadosi, S.T., and Aslin, R.N. (2014). The Goldilocks effect in infant auditory attention. *Child Development*, 85(5):1795–1804.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, Canada.

Kohonen, T. (1989). *Self-organization and associative memory*, New York: Springer Berlin Heidelberg.

Kool, W., Gershman, S.J., and Cushman, F. (2018). Planning complexity registers as a cost in metacontrol. *Cognitive Neuroscience*, 30(10):1391–1404.

Koulakov, A.A. and Chklovskii, D.B. (2001). Orientation preference patterns in mammalian visual cortex: a wire length minimization approach. *Neuron*, 29(2):519–527.

Kulkarni, T.D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain, pages 3675–3683.

Kulkarni, T.D., Saeedi, A., Gautam, S., and Gershman, S.J. (2016). Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Lange, S. and Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Barcelona, Spain, pages 1–8.

Lange, S., Riedmiller, M., and Voigtlander, A. (2012). Autonomous reinforcement learning on raw visual input data in a real world application. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Brisbane, Australia, pages 1–8.

Lee, S.W., Shimojo, S., and O'Doherty, J.P. (2014). Neural computations underlying arbitration between model-based and model-free learning. *Neuron*, 81(3):687–699.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373.

Levine, S., Wagener, N., and Abbeel, P. (2015). Learning contact-rich manipulation skills with guided policy search. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, United States, pages 156–163.

Levy, A., Platt, R., and Saenko, K. (2019). Hierarchical reinforcement learning with hindsight. In *Proceedings of the International Conference of Learning Representations (ICLR)*, New Orleans, LA, United States.

Lieder, F. and Griffiths, T.L. (2015). When to use which heuristic: A rational solution to the strategy selection problem. In *the 37th Annual Conference of the Cognitive Science Society*, pages 1362–1367.

Lihong, L., Littman, M.L., and Mansley, C.R. (2009). Online exploration in least-squares policy iteration. In *The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Budapest, Hungary, pages 733–739.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Luciw, M., Graziano, V., Ring, M., and Schmidhuber, J. (2011). 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, Germany, pages 1–8.

Machado, M.C., Bellemare, M.G., and Bowling, M. (2019). Count-based exploration with the successor representation. *arXiv preprint arXiv:1807.11622*.

Mahadevan, S. (2018). Imagination machines: A new challenge for artificial intelligence. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 7988–7993.

Mannella, F., Santucci, V.G., Somogyi, E., Jacquey, L., O'Regan, K., and Baldassarre, G. (2018). Know your body through intrinsic goals. *Frontiers in Neurorobotics*, 12:30.

Martinetz, T. and Schulten, K. (1991). A" neural-gas" network learns topologies. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, Helsinki, Finland, pages 397–402.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, New York, United States, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Moerland, T.M., Broekens, J., and Jonker, C.M. (2017). Efficient exploration with Double Uncertain Value Networks. In *Deep Reinforcement Learning Symposium at* *the Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, United States.

Mohamed, S. and Rezende, D.J. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, Montréal, Canada, pages 2116–2124.

Moulton, S.T. and Kosslyn, S.M. (2009). Imagining predictions: mental imagery as mental emulation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1521):1273–1280.

Nagabandi, A., Kahn, G., Fearing, R.S., and Levine, S. (2018). Neural networks dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, pages 7559–7566.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In *Advances In Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, pages 4026–4034.

Ostrovski, G., Bellemare, M.G., Oord, A.V.D., and Munos, R. (2017). Count-based exploration with neural density models. In *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, Australia, pages 2721–2730.

Oudeyer, P.Y. and Kaplan, F. (2007). In search of the neural circuits of intrinsic motivation. *Frontiers in Neuroscience*, 1(1):225–236.

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

Özgür, Ş. and Barto, A.G. (2006). An intrinsic reward mechanism for efficient exploration. In *International Conference on Machine learning (ICML)*, Pittsburgh, Pennsylvania, USA, pages 833–840.

Pathak, D., Agrawal, P., Efros, A.A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the International Conference on Machine Learning (ICML),* Sydney, Australia, pages 2778–2787.

Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P.Y. (2018). Unsupervised learning of goal spaces for intrinsically motivated goal exploration. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada.

Pong, V., Gu, S., Dalal, M., and Levine, S. (2018). Temporal difference models: Model-free deep RL for model-based control. In *Proceedings of the International Conference of Learning Representations (ICLR),* Vancouver, BC, Canada.

Precup, D. (2000). *Temporal abstraction in reinforcement learning*, University of Massachusetts Amherst.

Ptak, R., Schnider, A., and Fellrath, J. (2017). The dorsal frontoparietal network: A core system for emulated action. *Trends in Cognitive Sciences*, 21(8):589–599.

Racanière, S., Weber, T., Reichert, D.P., Buesing, L., Guez, A., Rezende, D.J., Badia, A.P., Vinyals, O., Heess, N., Li, Y., and Pascanu, R. (2017). Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS),* Long Beach, CA, United States, pages 5690–5701.

Rohmer, E., Singh, S.P., and Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Tokyo, Japan, pages 1321–1326.

Rohmer, E., Singh, S.P., and Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1321-1326.

Russek, E.M., Momennejad, I., Botvinick, M.M., Gershman, S.J., and Daw, N.D. (2017). Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS Computational Biology*, 13(9):e1005768.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *Proceedings of the International Conference of Learning Representations (ICLR)*, San Juan, Puerto Rico.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Shelhamer, E., Mahmoudieh, P., Argus, M., and Darrell, T. (2017). Loss is its own reward: Self-supervision for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, Toulon, France.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. (2018). Universal planning networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden, pages 4732–4741.

Stadie, B.C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Stahl, A.E. and Feigenson, L. (2017). Expectancy violations promote learning in young children. *Cognition*, 163:1–14.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. (2018). Intrinsic motivation and automatic curricula via asymmetric self-play. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada.

Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh International Conference on Machine Learning (ICML)*, Austin, TX, United States, pages 216–224.

Sutton, R.S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Talvitie, E. (2017). Self-correcting models for model-based reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, United States, pages 2597–2603.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. (2017). # Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, United States, pages 2750–2759.

Twomey, K.E. and Westermann, G. (2015). 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, pages 1–6.

Van Hasselt, H. (2012). Reinforcement learning in continuous state and action spaces. In *Reinforcement Learning*, Springer, Berlin, Heidelberg, pages 207–251.

Van Hasselt, H. and Wiering, M.A. (2007). Reinforcement learning in continuous action spaces. In *Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279.

Wiskott, L. and Sejnowski, T. (2002). Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14:715–770.

Xu, T., Liu, Q., Zhao, L., and Peng, J. (2018). Learning to explore via meta-policy gradient. In *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholm, Sweden, pages 5459–5468.

Yousefi, B. and Loo, C.K. (2016). Slow feature action prototypes effect assessment in mechanism for recognition of biological movement ventral stream. *International Journal of Bio-Inspired Computation*, 8(6):410–424.

Zafeiriou, L., Nicolaou, M.A., Zafeiriou, S., Nikitidis, S., and Pantic, M. (2013). Learning slow features for behaviour analysis. In *Proceedings of the IEEE*

International Conference on Computer Vision (ICCV), Sydney, Australia, pages 2840–2847.

Zhang, Z. and Tao, D. (2012). Slow feature analysis for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:436–450.

Zhou, X., Weber, C., and Wermter, S. (2017). Robot localization and orientation detection based on place cells and head-direction cells. In *Proceedings of the 26th International Conference on Artificial Neural Networks (ICANN)*, Sardinia, Italy, pages 137–145.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum Unterschrift