CURIOUS: Intrinsically Motivated Multi-Task, Multi-Goal Reinforcement Learning

Cédric Colas
INRIA - Flowers Team
Bordeaux, France

Olivier Sigaud
Sorbonne University - ISIR
Paris, France

Pierre-Yves Oudeyer
INRIA - Flowers team
Bordeaux, France

Abstract

In open-ended and changing environments, agents face a wide range of potential tasks that may not come with associated reward functions. Such autonomous learning agents must be able to generate their own tasks, some of which might prove easy, others impossible. For this reason, they should be able to actively select which task to attend at any given moment, to maximize their overall mastery on the set of learnable tasks. This paper proposes CURIOUS, a modification of Universal Value Function Approximators that enables intrinsically motivated agents to learn to achieve both multiple tasks and multiple goals within a unique policy. Agents focus on achievable tasks first, using an active learning mechanism that biases their attention towards tasks maximizing the absolute learning progress. This mechanism provides robustness to catastrophic forgetting and distracting tasks.

1 Introduction

In *autonomous continual learning*, agents must evolve in an open-ended, changing world and might face a variety of potential tasks. In such realistic environments, tasks cannot always be pre-specified by engineers. In situations where the reward is sparse, deceptive, or even non-existing, agents must be endowed with intrinsic motivations to explore the possibilities offered by their environments. They must set their own tasks and goals, and generate their own curriculum to practice them. This challenge can be approached within the framework of Intrinsically Motivated Goal Exploration Processes [1], leveraging computational models of autonomous development in human infants [2].

A task is defined as a set of constraints to be satisfied and is characterized by a reward function. Some tasks can be parameterized by goals e.g., if the task is to place a cube on a target, the position of this target can be seen as a goal. While multi-goal [3; 4; 5] and multi-task [6; 7] settings have been explored separately, only few works deal with both simultaneously [8]. Here we present CURIOUS[1], a multi-task, multi-goal reinforcement learning algorithm that uses intrinsic motivations to share learning across different tasks. To build such an algorithm, one must address three problems: 1) How to choose the policy architecture? 2) How to transfer knowledge efficiently between tasks and goals? 3) How to select the next task and goal to attend?

Related work. Kaebling et al. (1993) proposed the first algorithm able to leverage cross-goal learning to address different goals among a finite set [9]. For each possible goal, the algorithm learned a specific policy and its associated value function using Q-learning (*goal-experts* approach). More recently, Schaul et al. (2015) proposed Universal Value Function Approximators (UVFAs) [4]. A unique policy can address an infinity of goals by concatenating the current state and goal to feed both the policy and the value function (*multi-goal* approach). In UNICORN, UVFAs are used to learn

Contacts: firstname.lastname@{inria, upmc, inria}.fr

[1] CURIOUS stands for Continual Universal Reinforcement learning with Intrinsically mOtivated sUbstitutionS.

Preprint. Work in progress.

several tasks in parallel: reaching different objects in a visual world (*multi-task* approach) [6]. These works can be considered either as multi-goal or multi-task learning. Finally, Forestier et al. (2016) proposed an algorithm achieving both multi-task and multi-goal learning using a population-based algorithm that mutates and replays controllers experienced in the past [8].

These ideas prove better than simply training a policy per task/goal because knowledge can be transferred between different tasks/goals using off-policy and hindsight learning. Off-policy learning enables the use of any transition to improve the current policy: transitions collected from older policies [10], exploratory policies [11], or demonstrations [12]. Transitions collected while aiming at a particular task or goal can therefore be reused to learn about any other. When the set of goals/tasks is finite [6; 9], each transition is generally used to update the policy on every other goal/task respectively. When the goal space is continuous, *goal substitutes* are sampled at random [4; 5]. In UVFAs, this consists in the substitution of the goal or task that is part of the input, a technique called *goal/task-replay* or *goal/task-substitution*. Andrychowicz et al. (2017) proposed HER, a related idea for transferring knowledge between goals [5]. For each transition, the original goal can be substituted by any outcome experienced later in the same trajectory. This helps to increase the probability to observe rewards in reward-sparse environments. For a particular transition and after a goal or a task substitution, the reward must be re-evaluated.

Forestier et al. (2016) biased the selection of the next task to attempt towards the one showing the highest absolute measure of learning progress [8]. This mechanism helps the agent to engage less in tasks that are impossible or already solved, and to focus on achievable ones.

Another line of work implements multi-task learning and considers all the tasks but the most difficult as *auxiliary tasks*. In SAC-X, the idea is to increase the probability to observe rewards for the difficult task (placing cubes inside a box) by training on a set of simpler tasks [7]. In their experiment, one network is used for each task, and only the samples are shared (*task-expert* approach).

Contributions. The contributions of this paper are: 1) An extension of UVFAs to consider multiple tasks and goals within a single policy; 2) A smarter replay policy than random when the agent faces tasks that: a) are already solved; b) are too difficult; c) change in difficulty. The agent should not spend too much time learning about an already solved task, or an impossible one. If a task is solved, but somehow becomes more difficult (e.g. because part of the system broke), the agent should reallocate time to learn it again. We propose an extension of the idea of cross-task learning presented in UNICORN by selecting the substitute task so as to maximize an absolute measure of the learning progress (LP). 3) A new environment for multi-task and multi-goal learning. 4) Empirical comparisons to other architectures and study of the different learning phases demonstrated by our algorithm. 5) Properties of resistance to *distracting tasks* and *catastrophic forgetting*.

2 A Multi-Task Multi-Goal Environment

Multi-Task Fetch Arm is a multi-task and multi-goal environment based on environments from OpenAI Gym [13]. A robotic arm faces five cubes randomly positioned on a table, three of which are out of reach. The agent can target one of $N = 7$ tasks: (T_1) reaching a 3D target with the gripper; (T_2) reaching a 2D target on the table with cube 1; (T_3) reaching a 3D target with cube 1 above cube 2; (T_4) stacking cube 1 over 2; (T_{5-7}) reaching a 2D target on the table with the out-of-reach and randomly moving cubes 3 to 5 respectively (distracting tasks). We call *outcome* the position of the object specified by the current task at a given time. It is the result of trying to reach the *goal*. The reward function $R_{T,g}$ is sparse, binary and parameterized by both the task and goal. See Supplementary Materials for further details.

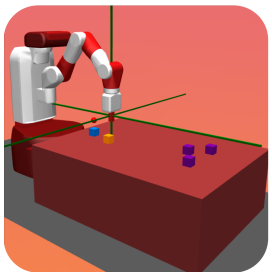


Figure 1: The *Multi-Task Fetch Arm* environment

3 CURIOUS: Intrinsically Motivated Multi-Task, Multi-Goal Learning

A multi-task, multi-goal architecture using universal approximators. UVFAs concatenate the agent's goal with its current state to form the input of the policy and the value function [4]. We

propose CURIOUS, an extension of UVFAs to enable multiple tasks and goals learning within a single network (*multi-task, multi-goal* approach). Given the goal space $\mathcal{G}_{T_i}$ of the current task T_i, the current goal g is defined by a vector of size $\sum_{i=0}^{N} dim(\mathcal{G}_{T_i})$. The goal g is set to 0 everywhere except in the indices corresponding to T_i, where it is set to g_i. By masking the inputs corresponding to unconsidered tasks, the corresponding weights are frozen during backpropagation. In addition, a task descriptor $task_d$ of size N is built to encode the current task (one-hot encoding). The overall input to the policy network is $[s_t, g, task_d]$, see Supplementary Figure 1. The task T_i remains constant for the entire episode.

Cross-task and cross-goal learning. In UNICORN, all transitions are replayed with all possible tasks [6]. However, not all tasks are equivalent. Two main cases advocate for a smarter replay policy: 1) in presence of a distracting task, there is nothing to be learned by wasting resources on replaying a transition for that task; 2) when the task is already learned, the agent should engage less in this task. We propose to use learning progress [14] to guide the selection of the task to replay, in a similar way as [8]. Here, the agent focuses its attention on tasks for which it is making the largest absolute progress, and pays little attention to tasks that are already solved or unsolvable, i.e. for which learning progress stays small. Taking the absolute value of the learning progress also leads to prioritize tasks for which the agent is showing decreasing performances. This helps to deal with catastrophic forgetting: the agent reallocates learning resources to the tasks it is forgetting. As in [8], the agent's competence on each task is evaluated as the success rate computed over a time window of training episodes. The asbolute learning progress (LP) is computed as the difference between present and past measures of competence. The agent selects the substitute task at random with probability $\epsilon = 0.4$ for exploration purposes and uses probabilities proportional to the learning progress otherwise (p_{LP}). Cross-goal learning is achieved using the HER replay strategy [5]. See Supplementary Materials for details about the computation of competence, LP and p_{LP}.

Task and goal selection. In these experiments, the next task T_i is sampled from the set of tasks $\mathcal{T}$ using p_{LP}, and the goal is sampled uniformly inside the corresponding goal space $\mathcal{G}_{T_i}$.

4 Experiment and Results

Experiments. In this paper, we present a set of experiments comparing: 1) A *multi-goal uni-task* architecture where goals are selected inside a holistic goal space including all tasks. This goal-parameterized architecture is equivalent to the Hindsight Experience Replay algorithm (HER); 2) A *multi-goal task-experts* architecture (MG-TE) where an expert multi-goal policy is trained for each of the N tasks. Each policy is trained one epoch every N on its designated task and shares its transitions with the others. When evaluated on a particular task, the algorithm uses the corresponding task-expert; 3) A *multi-task, multi-goal* architecture with intrinsically motivated task replay (CURIOUS). This approach uses a policy parameterized by the current task and goal. It selects both the next task to attempt and the substitute task using probabilities biased by LP, p_{LP}.

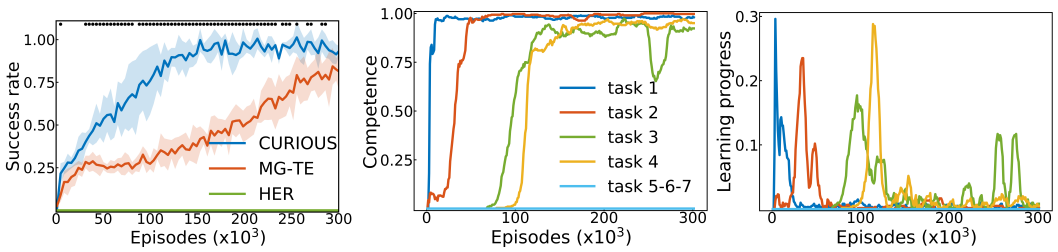


Figure 2: a: Average success rates computed over achievable tasks (4/7). Mean +/- std over 5 trials are plotted, while dots indicate significance when testing CURIOUS against MG-TE with a Welch's t-test at level $\alpha = 0.01$ (one-tail). b: Task-dependent measures of competence for CURIOUS (1 run). c: Task-dependent measures of absolute learning progress corresponding to (b) for CURIOUS.

Results. Figure 2a shows the evolution of the average success rate computed w.r.t. achievable tasks (4/7) for the 3 algorithms. The learning curve of HER stays flat. This can be easily understood, as none of the goals expressed in the full multi-task goal space corresponds to a real situation (e.g. the agent cannot reach the target with its gripper while staying away from it to get the reward corresponding to the stacking task). This motivates the use of a modular representation with separated tasks for an autonomous agent. Comparing MG-TE and CURIOUS, we can see that the achievable tasks are learned much faster in the multi-task and multi-goal approach ($\sim 150 \cdot 10^3$ vs. $\sim 300 \cdot 10^3$ episodes).

Figure 2b shows the evolution of the competence for each task using the CURIOUS algorithm for a particular trial, while Figure 2c shows the evolution of the corresponding LP measures. These figures demonstrate the existence of successive learning phases, that can be interpreted as developmental phases [2]. The robot first learns how to control its gripper (T_1), then to push objects on desired target on the table (T_2) before it learns how to place the cube on a 3D target (T_3) and how to stack the two cubes (T_4). These two last tasks are not always learned in the same order. Once the agent has experienced a few successes on one of these tasks, it focuses on it, which leads to more successes. Noise in the individual experience triggers different developmental trajectories depending on the individual (all competence figures can be found in the Supplementary Materials). There is no progress to be made on task (T_{5-7}) as the distracting cubes 3 to 5 cannot be reached. Figure 2c shows that LP stays small for tasks that are already solved (e.g. T_1 after 10^4 episodes) or unsolvable (e.g. T_{5-7}), and increases when the tasks are being learned.

Looking at Figure 2b, we can observe a drop in the competence on T_3 around episode $250 \cdot 10^3$. This phenomenon is usually described as catastrophic forgetting: while training on the other tasks, the network forgets about T_3, that was previously mastered. The corresponding period of Figure 2c shows an increase in LP for T_3, which in turn triggers an additional focus of the agent towards that task. Using LP to bias its attention, the agent monitors its competence on the tasks and can react when it forgets about a previously mastered task. This mechanism helps fighting the problem of catastrophic forgetting and facilitates learning of multiple tasks in parallel.

5 Conclusion

This paper has proposed CURIOUS, an extension of UVFAs to enable multi-task and multi-goal learning in a single policy. Active mechanisms bias the agent's attention towards tasks where the absolute learning progress is maximized. This induces distinct learning phases, some of which are shared across agents, others depending on the agent experience. With this mechanism, agents spend less time on impossible tasks and focus on achievable ones. This is important for continual learning in the real world, where agents set tasks to themselves and might face tasks with diverse levels of difficulty, some of which might be required to solve others later on. This mechanism also enables to fight the catastrophic forgetting issue, by refocusing learning on tasks that are being forgotten.

As noted in [6], representations of the world state are learned in the first layers of a policy neural network. A representation learned for one task could probably be useful for another similar one. Our muti-task, multi-goal policy leverages that fact, by re-using subparts of the same network to learn different but similar tasks. This might partially explain why our *multi-task and multi-goal* approach outperforms the *multi-goal task-experts* (MG-TE) policy architecture (Figure 2a), although further work should investigate the relative contributions of the policy architecture and the active mechanisms for task selection and replay.

Future Work. Future experiments will study the impact of changes in the learning setting (e.g. agent's physical properties, dynamic changes of task-set). Although the task-set is pre-defined by the engineer, we do not consider them as coming from the environment. To learn autonomously, agents must be able to construct their own. This vision comes from the IMGEP framework [1], which defines agents able to set their own goals to explore and master their environment. Our algorithm can be seen as a monolithic implementation of such algorithms. Further work will aim at combining CURIOUS to the autonomous learning of task sets and goal spaces using representation learning [15].

Links. The environment is available online at `https://github.com/qdrn/gym_flowers`. A video of the results can be seen at `https://frama.link/CURIOUS_results`, our code will soon be released.

Acknowledgments

Cédric Colas is partly funded by the French Ministère des Armées - Direction Générale de l'Armement. Olivier Sigaud is partly funded by the European Commission, within the DREAM project. The DREAM project has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement N^o 640891.

Parallel Submissions

This work has also been submitted to the NIPS Deep RL workshop, 2018.

References

[1] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[2] Pierre-Yves Oudeyer and Linda B Smith. How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, 8(2):492–502, 2016.

[3] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[4] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[6] Daniel J Mankowitz, Augustin Žídek, André Barreto, Dan Horgan, Matteo Hessel, John Quan, Junhyuk Oh, Hado van Hasselt, David Silver, and Tom Schaul. Unicorn: Continual learning with a universal, off-policy agent. *arXiv preprint arXiv:1802.08294*, 2018.

[7] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[8] Sébastien Forestier and Pierre-Yves Oudeyer. Modular active curiosity-driven discovery of tool use. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3965–3972. IEEE, 2016.

[9] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018.

[12] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*, 2017.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[14] Frédéric Kaplan and Pierre-Yves Oudeyer. Maximizing learning progress: an internal reward system for development. In *Embodied artificial intelligence*, pages 259–270. Springer, 2004.

[15] Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. In *Proceedings of CoRL 2018 (PMLR)*, 2018.

[16] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9(5):1054–1054, 1998.

[17] Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

6 Supplementary Material

6.1 Supplementary Background

Reinforcement Learning. We consider a Reinforcement Learning (RL) problem [16] based on a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, p is the transition probability between states $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, r is the reward function $\mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ and γ is the discount factor. In our case, the reward function is not considered external but internal. The policy is a function mapping the current state to the next action, $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The optimal Q-value function Q^* is the expected value of the γ-discounted sum of rewards the agent can experience from its current state s after performing action a and following the optimal policy π^* thereafter: $Q^*(s, a) = E_{\pi^*}\big(\sum_{\tau=0}^{\infty} \gamma^\tau r^{t+\tau} | s_t = s, a_t = a\big)$. In deep Reinforcement Learning (deep RL), the Q-function and the policy are approximated by deep neural networks.

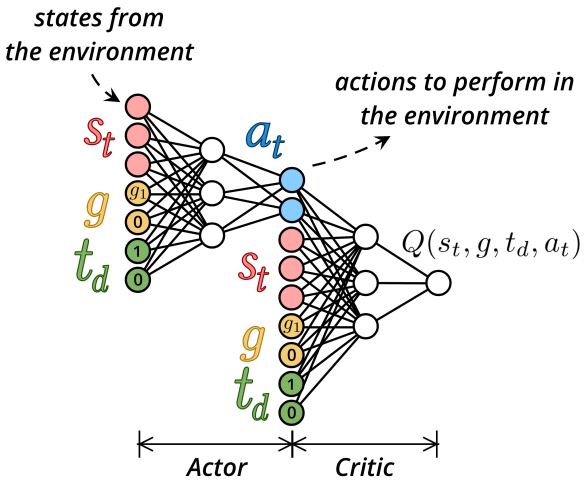
Universal Value Function Approximators. UVFAs parameterize the policy and the value function by a goal [4]. In practice, for neural networks function approximators, this consists in the concatenation of the state and the goal to form the network input $Q(s, a, g)$ and $\pi(s, g)$. This way, a single policy can be used to address any goal.

6.2 Supplementary Methods

Environment details. *Multi-Task Fetch Arm*[2] is a new multi-task and multi-goal simulated environment based on the Fetch environments included in the OpenAI Gym suite [13]. The agent is embodied by a 7-DoF Fetch robotic arm facing five randomly positioned cubes on a table, three of which are out of reach. The agent controls the 3D Cartesian position of its gripper in velocity as well as its two-fingered parallel gripper. The agent can target one of $N = 7$ tasks: (T_1) reaching a 3D target with the gripper; (T_2) reaching a 2D target on the table with cube 1; (T_3) reaching a 3D target with cube 1 above cube 2; (T_4) stacking cube 1 over cube 2; (T_{5-7}) reaching a 2D target on the table with the out-of-reach and randomly moving cubes 3 to 5 respectively (distracting tasks). The reward function $R_{T,g}$ is sparse, binary and parameterized by both the task and the goal. The reward is 0 when the Euclidean distance between the considered object and the goal is less than $\epsilon = 0.05$, -1 otherwise. The stacking task T_4 has an additional constraint and provides a reward when the gripper is far from the stacked cube. The observation space has 49 dimensions while the action space has 4 (3D actions + gripper).

[2] Available online at https://github.com/qdrn/gym_flowers

The actor-critic architecture. Supplementary Figure 3a, represents the actor-critic architecture. We extend the idea of UVFAs by concatenating a task descriptor encoding the current task to the input of the actor and critic. The full input fed to the actor and critic is therefore $[s_t, g, task_d]$. We call this task- and goal-parameterized architecture Extended-UVFAs (E-UVFAs). The actor implements the policy and maps the concatenation of the current state, the episode goal and the task descriptor $[s_t, g, task_d]$ to the next action a_t. This action vector is then concatenated to a copy of the actor's input to feed the critic $[s_t, g, task_d, a_t]$. The critic provides an approximate of the Q-value: $Q(s_t, g, task_d, a_t)$. Critic and actor are then trained using DDPG [10], although any other off-policy actor-critic method could be used (e.g. TD3 [17]).



a

Figure 3: **Task- and goal-parameterized actor-critic RL architecture (E-UVFAs)** with 2 tasks parameterized by one dimensional goals g_i each. The agents is attempting goal g_1 in task T_1, as specified by the task descriptor $t_d = [1, 0]$. The actor (left) computes the action a_t. The critic (right) computes the Q-value.

Tracking competence and absolute learning progress. The agent needs to keep track of its competence and learning progress for each task. To do this, it evaluates itself on random tasks and goals for one rollout every 10, without exploration noise. The results (success 1 or failure 0) of these rollouts are stored in competence queues. As in Forestier et al. (2016), the agent's competence on a task is then computed as the average over the last 300 results recorded for that task [8]. The absolute learning progress (LP) is computed as the absolute value of the difference between the current competence and the one measured 300 recorded rollouts before. LP is used for two purposes: 1) biasing the selection of the next task to try (*task selection*); 2) biasing the selection of the task to substitute (to replay) in the next minibatch used for training the policy and the value function (*task-replay* or *task-substitution*). Both cases can be represented as a stochastic multi-arm bandit problem, where the agent needs to repeatedly select tasks from a finite task-set $\mathcal{T}$ in order to maximize the absolute learning progress. Here, we implement a simple approach called proportional probability matching, with an additional ϵ-greedy strategy for exploration. We compute the *learning progress probabilities* $p_{LP}(T_i)$ as:

$$p_{LP}(T_i) = \epsilon \times \frac{1}{N} + (1 - \epsilon) \frac{LP(T_i)}{\sum_{j=1}^{N} LP(T_j)},$$

where $LP(T_i)$ is the absolute learning progress of T_i and N is the number of tasks. The ratio ϵ ($\epsilon = 0.4$) implements a mixture between random exploration of tasks (left term) and biased selection/replay of tasks (right term). The random exploration term enables to sample tasks that do not show any learning progress (i.e. already solved, not solved, or at a plateau). This way, the agent can check that it stays competent on tasks that are already learned, or insist on tasks that are currently too hard. Once the task and goal have been substituted, the internal reward must be computed again using the substitute task and goal.

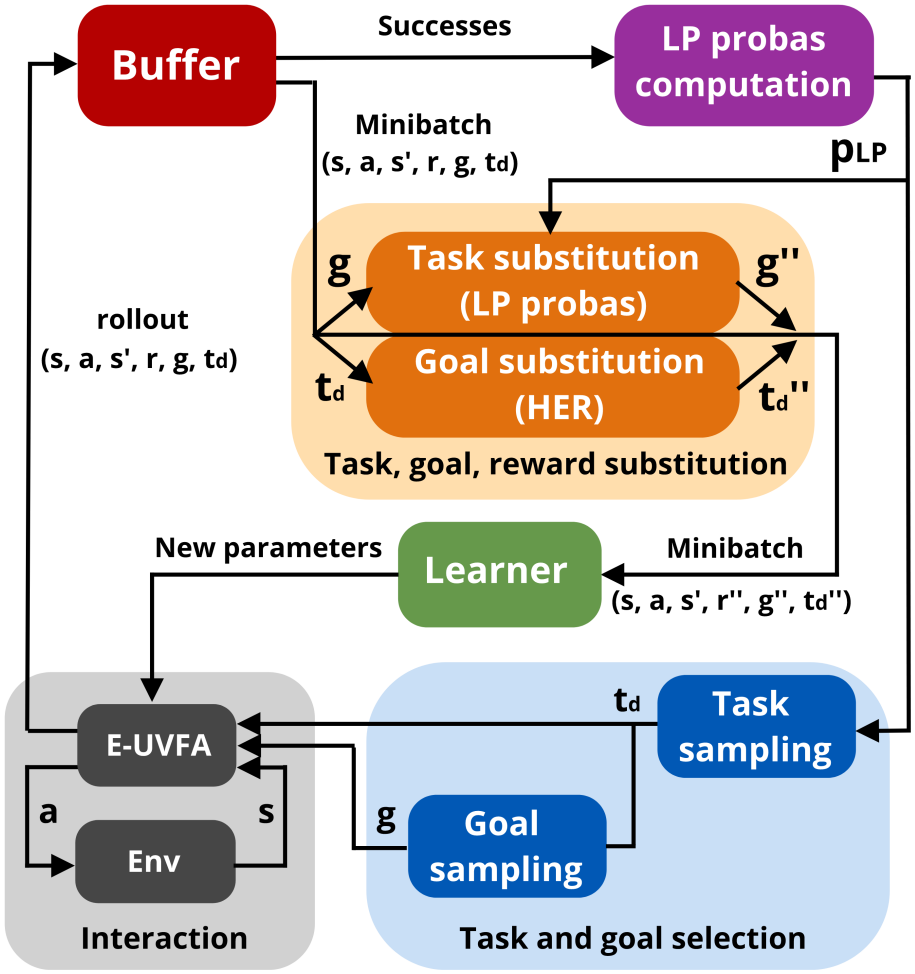


Figure 4: **Schematic view of CURIOUS.** LP stands for absolute learning progress.

Implementation. The pseudo-code is detailed in Algorithm 1 and represented in Supplementary Figure 4. First, the task and goal for the next rollout should be selected (in blue in Supplementary Figure 4). The task selection follows the learning progress probabilities p_{LP} (in purple). The goal selection is implemented by a uniform sampling inside the goal space corresponding to the selected task $\mathcal{G}_{T_i}$. To update the policy and critic, the algorithm samples a minibatch from the replay buffer (red) and implements task and goal substitutions to perform cross-task and cross-goal learning (orange). For each transition, the substitute task is selected following p_{LP}, whereas the substitute goal is selected using the hindsight strategy proposed in [5]. This means that the goal is sometimes ($p = 0.8$) replaced by an outcome reached later in the same episode. A new reward is computed for each transition, given the internal reward function corresponding to the substitute task and goal. Finally, the policy update is conducted by the learner (green). The CURIOUS algorithm is built on top of the OpenAI Baselines implementation of HER-DDPG.[3] It uses the same hyperparameters except for the number of training iterations per epoch (100 in our case) [3]. Note that this consists in a parallel implementation with 19 actors. The actors share the same parameters and their updates are averaged to compute the next set of parameters.

[3] The OpenAI Baselines implementation of HER can be found at https://github.com/openai/baselines, ours will soon be released.

Algorithm 1 The CURIOUS algorithm

Input: env, $\mathcal{T}$, $\mathcal{G}_{1:N}$, $noise$, internal_reward( ) ▷ $\mathcal{T}$: set of N tasks, $\mathcal{G}_i$: goal space of task T_i
Initialize: $policy, memory, p_{LP}$
while learning not done **do**
  $goal, task_d(T_i) \leftarrow$ **Task-GoalSelector()** ▷ $T_i \sim p_{LP}$, $goal \sim \mathcal{U}(\mathcal{G}_i)$
  for $t = 0 : N_t$ **do**
    $s_t \leftarrow$ env.reset()
    $policy\_input \leftarrow$ concatenate$(s_t, task_d, goal)$
    $a_t \leftarrow policy(policy\_input)$
    $a_t \leftarrow a_t + noise$ ▷ Unless the agent is evaluating its competence
    $s_{t+1} \leftarrow$ env.step(a_t)
    $r_t \leftarrow$ internal_reward( ) ▷ r_t is computed internally
  $memory$.add$(s_t, a_t, s_{t+1}, r_t, goal, task_d)$
  $p_{LP} \leftarrow memory$.compute_proba_progress( )
  $batch \leftarrow memory$.sample( )
  $modified\_batch \leftarrow$ **Task-GoalReplayPolicy(**$batch$, p_{LP}**)** ▷ Use p_{LP} and HER, new r_t
  $policy \leftarrow$ **PolicyUpdate(**$modified\_batch$**)** ▷ With DDPG

6.3 Supplementary Results

In Supplementary Table 1, we present all the competence and LP curves for the 5 trials of algorithm CURIOUS. We can see that T_1 (reaching a target with the gripper) is always learned first and T_2 (pushing cube 1 over a target) is always learned second. After that, T_3 and T_4 can be learned in various order or even simultaneously depending on the individual learning trajectories (e.g. trial 1). Indeed, when a few rewards are collected for a task, LP increases. This leads to additional focus towards that task, which generates even more progress. What happened by chance during the initial learning phases of the agent leads this agent to focus first on either T_3 or T_4. Although some tasks might be easier to learn first, or necessary to learn others, individual experience can influence learning trajectories just as for humans [2].

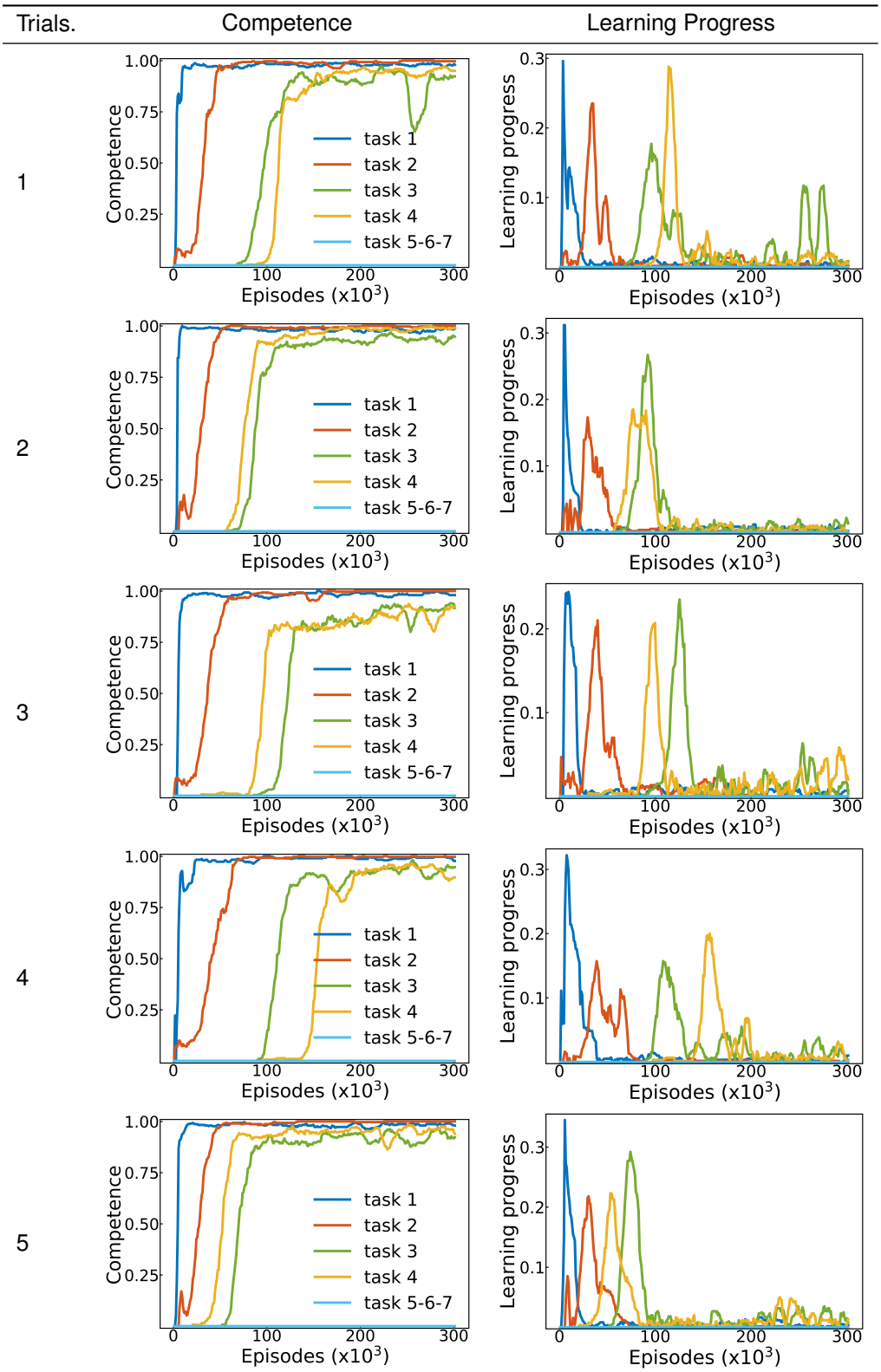


Table 1: Competence (left) and absolute learning progress (right) for the 5 trials of the CURIOUS algorithm.