



# Skill-based curiosity for intrinsically motivated reinforcement learning

Nicolas Bougie<sup>1,2</sup> · Ryutaro Ichise<sup>1,2</sup>

Received: 26 April 2019 / Revised: 9 July 2019 / Accepted: 16 September 2019 / Published online: 10 October 2019  
© The Author(s) 2019

## Abstract

Reinforcement learning methods rely on rewards provided by the environment that are extrinsic to the agent. However, many real-world scenarios involve sparse or delayed rewards. In such cases, the agent can develop its own intrinsic reward function called curiosity to enable the agent to explore its environment in the quest of new skills. We propose a novel end-to-end curiosity mechanism for deep reinforcement learning methods, that allows an agent to gradually acquire new skills. Our method scales to high-dimensional problems, avoids the need of directly predicting the future, and, can perform in sequential decision scenarios. We formulate the curiosity as the ability of the agent to predict its own knowledge about the task. We base the prediction on the idea of skill learning to incentivize the discovery of new skills, and guide exploration towards promising solutions. To further improve data efficiency and generalization of the agent, we propose to learn a latent representation of the skills. We present a variety of sparse reward tasks in MiniGrid, MuJoCo, and Atari games. We compare the performance of an augmented agent that uses our curiosity reward to state-of-the-art learners. Experimental evaluation exhibits higher performance compared to reinforcement learning models that only learn by maximizing extrinsic rewards.

**Keywords** Reinforcement learning · Exploration · Autonomous exploration · Curiosity in reinforcement learning

## 1 Introduction

Combining reinforcement learning (RL) with neural networks has led to a large number of breakthroughs in many domains. They work by maximizing extrinsic rewards provided by

---

Editors: Kee-Eung Kim and Jun Zhu.

✉ Nicolas Bougie  
nicolas-bougie@nii.ac.jp

Ryutaro Ichise  
ichise@nii.ac.jp

<sup>1</sup> The Graduate University for Advanced Studies, Sokendai Tokyo, Japan

<sup>2</sup> National Institute of Informatics, Tokyo, Japan

the environment. In well-specified reward function tasks, such as Atari game (Abbeel et al. 2007) or robotic control (Levine et al. 2016), RL is able to achieve human performance. In reality, many domains involve reward functions that are sparse or poorly-defined.

Overcoming sparse reward scenarios is a major issue in RL since the agent can only reinforce its policy when the final goal is reached. In such scenarios, motivating the agent to explore the environment is necessary. Several works have attempted to facilitate exploration by combining the original reward with an additional reward. For instance, reward shaping (Ng et al. 1999) guides the policy optimization towards promising solutions with a supplementary reward, but, the necessity of engineering limits the applicability to specific domains. If we have demonstrations of the task, we can extract a reward function using inverse reinforcement learning (Ng and Russell 2000). However, such approaches are limited to behaviors that are easy for humans to demonstrate.

An alternative approach to address the reward design problem is to motivate the agent to explore novel states using curiosity—the ability to acquire new skills which might become useful in the future. Curiosity is motivated by human early stage of development: babies reward themselves for acquiring increasingly difficult new skills. Many formulations aim to measure the novelty of the states in order to encourage exploration of novel states. For instance, an approach (Racanière et al. 2017) bases the exploration of the agent on the surprise—the ability of the agent to predict the future. Nevertheless, these models remain hard to train in high dimensional state spaces such as images.

In this paper, we introduce a novel technique called goal-based curiosity (GoCu). We propose an alternative solution to the curiosity mechanism by generating an exploration bonus based on the agent’s knowledge about its environment. To do so, we use the idea of goals to automatically decompose a task into several easier sub-tasks, and, skills, the ability to achieve a goal given an observation. In summary, our main contribution is that we manage to bypass most pitfalls of previous curiosity-based works with the following idea: GoCu is not based on the ability to predict the future nor the novelty of the states, but on learning which skills are mastered. That is, often visiting a state will result in a high intrinsic reward unless the agent perceives that it has good knowledge about the associated skills. In details, given an observation the agent predicts which goals it masters. We then reward the agent when the uncertainty to achieve them is high, to encourage curiosity. Our method relies on two deep neural networks: (1) to embed the states and goals to a latent space and (2) a predictor network to predict the capabilities of the agent. In order to improve generalization among skills, we embed the goals to a latent space using a variational autoencoder (Kingma and Welling 2014).

We first evaluate our approach on a set of sequential sparse tasks (several sequential steps are required in order to achieve the final goal) in the Minigrid (Chevalier-Boisvert et al. 2018) environment. Next, we show that our approach can scale to large environments. Our method can also learn policies in the MuJoCo environment, as well as in several Atari games, in a small number of steps. We compare our algorithm against proximal policy optimization (PPO) (Schulman et al. 2017), and advantage actor-critic (A2C) (Mnih et al. 2016) agent as well as state-of-the-art curiosity-based RL methods. We show that curiosity-driven exploration is crucial in tasks with sparse rewards and demonstrate that our agent explores faster as compared to other methods.

## 2 Related work

The problem of motivating the agent to explore sparse reward environments was studied in the works that we discuss in this section. Most approaches can be grouped into three classes: reward shaping and auxiliary tasks, goal conditioned learning, and, intrinsic curiosity.

### 2.1 Reward shaping and auxiliary tasks

Reward shaping (Ng et al. 1999) aims to guide the agent towards potentially promising solutions with an additional hand-crafted reward. However, this idea is limited to specific domains by the necessity of human engineering. Other attempts have been made to design this additional reward without supervision. For instance, Stadie et al. (2015) improve exploration by estimating visitation frequencies for states and state-action pairs. When a state-action is not visited enough, a novelty bonus is assigned. Similarly, several works (Bellemare et al. 2016; Ostrovski et al. 2017) are based on a pseudo-count reward bonus to encourage the exploration of novel states. These methods assume that the states need to be equally visited. Obviously, this assumption is no longer valid in sequential tasks. By contrast, we let the agent learn which states need to be explored in priority. UNREAL (Jaderberg et al. 2017) trains the agent on unsupervised auxiliary tasks. In order to perform well, the agent must acquire a good representation of the states which entails that it must explore the environment. However, this method requires experts to design the auxiliary tasks. Our method enables the agent to discover sub-tasks without any prior belief about the target task.

### 2.2 Goal conditioned learning

Goal conditioned learning motivates an agent to explore by constructing a goal-conditioned policy and then optimize the rewards with respect to a goal distribution. For instance, the approach, universal value function approximators (Schaul et al. 2015), trains a goal conditioned policy but is limited to a single goal. Andrychowicz et al. propose an implicit curriculum by using visited states as target and improve sample efficiency by replaying episodes with different goals (Andrychowicz et al. 2017). However, selecting relevant goals is not easy. Another work (Florensa et al. 2017) proposes to generate a series of increasing distant initial states from a goal or to embed the goals to a latent space and then sample the goals (Held et al. 2017). Reinforcement Learning with Imagined Goals (RIG) (Nair et al. 2018) trains a policy to reach a latent goal sampled from a VAE. However, this formulation does not have a notion of which goals are hard for the learner. Instead, our method considers how difficult is to achieve a goal to help the agent to find more efficient solutions to reach it. Besides, we focus on the problem of optimizing learning of multiple skills simultaneously to improve data efficiency. Despite successes in robotic manipulation tasks, generating increasingly difficult goals is still challenging (Epepe et al. 2018), especially in sequential tasks. On the other hand, our method bypasses this issue by training the agent to gradually acquire new skills using multiple goals at once, and adapts curiosity based on the complexity of each state.

### 2.3 Intrinsic curiosity

Intrinsic curiosity refers to the idea of self-motivating the agent through an additional reward signal, in order to guide the agent towards the desired outcome. Most techniques can be grouped into two approaches.

The first approach to generate curiosity relies on estimating a novelty measure, in order to encourage the exploration of novel states. Schmidhuber et al. propose for the first time a curiosity reward proportional to the predictability of the task (Schmidhuber 1991a) and then, based on the learning progress (Schmidhuber 1991b). To give another example (Itti and Baldi 2006), the agent receives a reward proportional to the *Bayesian surprise*, the estimation of which data affect the prior belief about the world of the observer. Several other works focused on maximizing a measure such as empowerment (Salge et al. 2014) or, competence progress (Baranes and Oudeyer 2013). Recently, a solution was proposed to measure uncertainty to generalize count-based exploration (Bellemare et al. 2016). Nevertheless, these models are hard to train in high dimensional state spaces such as images, or, in stochastic environments.

The second approach consists in optimizing the ability of the agent to predict the consequences of its actions. Racanière *et al.* base the exploration of the agent on the surprise—the ability of the agent to predict future (Racanière et al. 2017). A related concept (Pathak et al. 2017) estimates the surprise of the agent by predicting the consequences of the actions of the agent on the environment. Namely, they use an inverse model to learn feature representations of controllable aspects of the environment. Nevertheless, predicting the future is unsuitable for domains in which variability is high, and tend to limit long-horizon performance due to model drift. In that work, the intrinsic curiosity combines agent’s experiences and a predictor to estimate how complex are the future states surrounding the current state; while embedding states to a latent space to improve generalization. Another line of work aims to predict the features of a fixed random neural network on the observation of the agent (Burda et al. 2018). However, the low sample efficiency does not show clearly how to adapt this method to large scale tasks. The intrinsic motivation may include the comparison between the current observation and the observations in an episodic memory (Savinov et al. 2019). This comparison uses the concept of reachability, but is limited to episodic tasks and does not use the idea of skill learning. Our work differs by proposing a technique that scales to continuous tasks and let the agent discover new skills and reinforce them based on their difficulty. Our solution is also more data efficient since we only need to do predictions on a small number of skills instead of on the full episodic memory. By embedding the skills and states, we also improve generalization capability of our agent. Finally, our method enables the use of efficient goal sampling strategies to reduce training time. A solution is to (Lehman and Stanley 2011) maintain a list of behaviors, and reward the agent for exploring novel behaviors. In this paper, we step toward this idea by considering the impact of the agent’s actions on its ability to achieve skills. Furthermore, only the agent’s experience is required to predict the uncertainty to master skills, thus alleviates the need of predicting complex changes in the environment.

### 3 Preliminaries

#### 3.1 Reinforcement learning

Reinforcement learning (Sutton 1988) consists of an agent learning a policy  $\pi$  by interacting with an environment. At each time-step the agent receives an observation  $s_t$  and chooses an action  $a_t$ . The agent gets a feedback from the environment called a reward  $r_t$ . Given this reward and the observation, the agent can update its policy to improve the future rewards.

Given a discount factor  $\gamma$ , the future discounted rewards, called return  $R_t$ , is defined as follows:

$$R_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'} \quad (1)$$

where  $T$  is the time-step at which the epoch terminates.

The agent learns to select the action with the maximum return  $R_t$  achievable for a given observation (Sutton and Barto 1998). From Eq. (1), we can define the action value  $Q^\pi(s, a)$  at a time  $t$  as the expected reward for selecting an action  $a$  for a given state  $s_t$  and following a policy  $\pi$ .

$$Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a] \quad (2)$$

The optimal policy  $\pi^*$  is defined as selecting the action with the optimal Q-value, the highest expected return, followed by an optimal sequence of actions. This obeys the Bellman optimality equation:

$$Q^*(s, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s', a') | s, a\right] \quad (3)$$

In temporal difference (TD) learning methods such as Q-learning or Sarsa, the Q-values are updated after each time-step instead of updating the values after each epoch, as happens in Monte Carlo learning.

### 3.2 Policy gradient reinforcement learning

Here we briefly introduce the policy gradient-based reinforcement learning methods. Instead of learning Q-values that represent the policy—the expected return given a state and action, policy gradient based methods learn directly a policy function. In our method, we use PPO (Schulman et al. 2017), a policy gradient based method, that was shown to have low oscillation during training, effective in high dimensional action spaces and can learn a stochastic policy.

The agent's policy maps a state to a probability distribution. This policy parameterized by  $\theta$  usually uses a neural network or a convolutional neural network (Krizhevsky et al. 2012) if inputs are images. We refer to  $\pi_\theta$  or  $J(\theta)$  as the value policy, the expected discounted rewards obtained by the agent when following the policy  $\pi_\theta$ .

$$J(\theta) = \mathbb{E}_\theta \left[ \sum_{t=0}^T \gamma^t r_t \right] \quad (4)$$

Sutton et al. (2000) demonstrate that the gradient of the value  $J$  can be estimated as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta [R(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t)] \quad (5)$$

with  $R(s_t, a_t)$  the return defined in Eq. 1. In order to reduce the variance of the gradient and to allow an update at each iteration, some methods such as proximal policy optimization (PPO) (Schulman et al. 2017) or trust region policy optimization (TRPO) (Schulman et al. 2015) use an estimation of the advantage function  $\hat{A}_t$ . The loss function to optimize becomes:

$$L^{PG}(\theta) = \mathbb{E}_t [A_t \log \pi_\theta(a_t | s_t)] \quad (6)$$

PPO introduces a penalty to control the change of the policy at each iteration to reduce oscillating behaviors. The novel objective function becomes:

$$L^{CLIP}(\theta) = \mathbb{E}_t [\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)] \quad (7)$$

where  $r_t$  is the ratio of the probability under the new and old policies and  $\epsilon$  a hyperparameter.

### 3.3 Variational autoencoders

Variational autoencoders (VAEs) (Fig. 1) have been widely used to learn latent representation of high dimensional data such as images (Kingma and Welling 2014). In this paper, we use a VAE to embed the observations from the environment. The input image is passed through an encoder network  $q_\phi$  which outputs the parameters  $\mu$  and  $\sigma$  of a multivariate Gaussian distribution. A latent vector is sampled and the decoder network  $p_\psi$  decodes it into the original state space. The parameters  $\phi$  and  $\psi$  of the encoder and decoder are jointly optimize to maximize the loss function:

$$\mathcal{L}(\psi, \phi; s^{(i)}) = -\beta D_{KL}(q_\phi(z|s^{(i)})||p(z)) + \mathbb{E}_{q_\phi(z|s^{(i)})} \left[ \log p_\psi(s^{(i)}|z) \right] \tag{8}$$

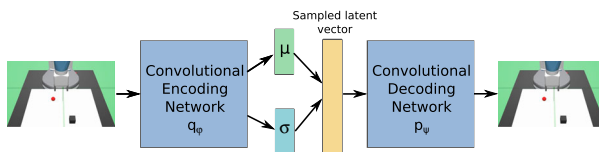
where the first term is a regularizer, the Kullback–Leibler divergence between the encoder distribution  $q_\phi(z|s^{(i)})$  and  $p(z)$ . This measure the divergence—how much information is lost when using q to represent p.  $p(z)$  is some prior specified as a standard normal distribution  $p(z) = \mathcal{N}(0, 1)$ . The second term is the expected negative log-likelihood—the reconstruction loss. The expectation is taken with respect to the encoder’s distribution over the representations.

## 4 Method

### 4.1 GoCu: goal-based curiosity algorithm

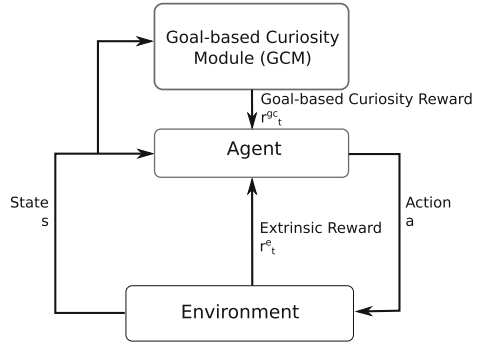
We consider an agent interacting with an environment; at each time step  $t$  the agent performs an action and receives an extrinsic reward  $r_t^e$  supplied by the environment. In sparse reward tasks, rewards are infrequent or delayed which entails that agents learn slowly.

We propose to introduce a new reward signal—goal-based curiosity reward  $r^{gc}$ , to encourage the agent to discover new skills (Fig. 2). In our approach, at time  $t$  the agent receives the sum of these two rewards  $r_t = r_t^e + r_t^{gc}$ . To encourage the agent to explore the environment and favor the discovery of new skills, we design  $r^{gc}$  to be higher in *complex* states. To do so, a curiosity signal is produced in novel states, and, in states involving skills that the agent seeks to reinforce.



**Fig. 1** Variational auto encoder structure. The input image is passed through an encoder network which outputs the parameters  $\mu$  and  $\sigma$  of a multivariate Gaussian distribution. A latent vector is sampled and the decoder network decodes it into an image

**Fig. 2** In a state  $s$ , the agent interacts with the environment by performing an action  $a$ , and receives an extrinsic reward  $r^e$ . A policy  $\pi(s_t; \theta_P)$  is trained to optimize the sum of  $r^e$  and  $r^{gC}$ . The intrinsic reward  $r^{gC}$  is generated by the goal-based curiosity module to favor the exploration of novel or complex states



**Algorithm 1** PPO + GoCu pseudo-code

- 1: Collect  $D = \{s^{(i)}\}$  using random exploration
- 2: Train VAE on  $D$  by optimizing (8)
- 3: Initialize replay buffer  $R$
- 4: Initialize goal buffer  $R_g$
- 5: Initialize environment env
- 6: Initialize goal-based curiosity module GCM
- 7:  $t = 0$
- 8: **for** rollout  $r=1$  to  $M$  **do**
- 9:     Sample  $K$  goals from  $R_g$   $g' = \{\phi(g) \sim f_P(g)\}$
- 10:    **for**  $l=1$  to  $N$  **do**
- 11:     Sample  $a_t \sim \pi(a_t|s_t)$
- 12:     Collect  $s_{t+1}, r_t^e \sim \text{env}(s_{t+1}, r_t^e|s_t, a_t)$
- 13:     Calculate curiosity-based reward  $r_t^{gC} = \text{GCM}(s_{t+1})$
- 14:     Add  $s_t, s_{t+1}, r_t^e, a_t, r_t^{gC}$  to  $R_r$
- 15:      $t+ = 1$
- 16:    Calculate returns  $R_{e,r}, R_{gC,r}$  for extrinsic and curiosity-based reward
- 17:    Calculate advantages  $A_{e,r}, A_{gC,r}$  for extrinsic and curiosity-based reward
- 18:    Calculate the new advantage  $A_r = A_{e,r} + \sigma A_{gC,r}$
- 19:    Optimize PPO using on  $R_r$  and  $A_r$
- 20:    Update GCM every  $K$  episodes on  $R$
- 21:    Add new goals to  $R_g$  given  $R_r$

The policy  $\pi(s_t; \theta_P)$  is represented by a deep neural networks. Its parameters  $\theta_P$  are optimized to maximize the following equation:

$$\max_{\theta_P} \mathbb{E} \pi(s_t; \theta_P) \left[ \sum_t r_t \right] \tag{9}$$

Any RL algorithm could be used to learn the policy  $\pi(s_t; \theta_P)$ . In this work, we use proximal policy optimization (PPO) (Schulman et al. 2017) as policy learning method. Our main contribution is the goal-based curiosity module (GCM) module that helps the agent in the quest of new knowledge. We describe more details of each step below. See Algorithm 1 for a more formal description of the algorithm. In order to combine the extrinsic  $r^e$  and the goal-based curiosity  $r^{gC}$  reward, we decompose the return into two terms  $R = R_e + R_{gC}$ . These two terms are estimated by collecting samples from the environment. The state values  $V_e$  and  $V_{gC}$  are computed by the same critic neural network with two heads, one for each value function. Finally, we combine them to obtain the advantage function:

$$A_r = A_{e,r} + \sigma A_{gC,r} \tag{10}$$

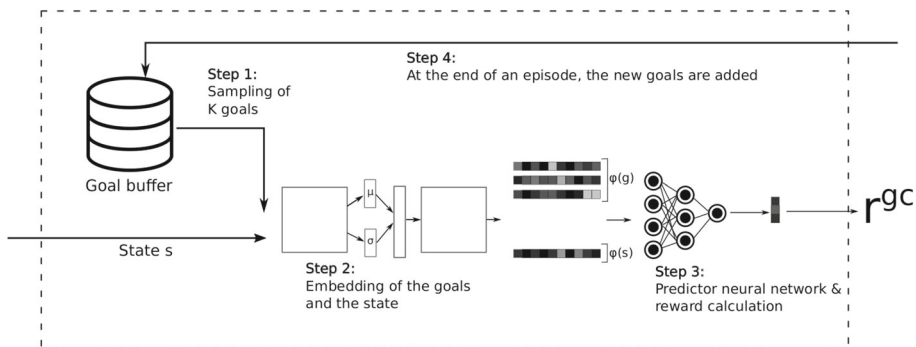
where  $\sigma$  controls the importance of the goal-based curiosity advantage. Note that the measure performance that we aim to improve with GoCu is the extrinsic rewards; the intrinsic rewards only intend to guide exploration of the environment.

### 4.2 Goal-based curiosity module (GCM)

We build the goal-based curiosity module (GCM) on the following idea. Rather than using count-based exploration, or next frame prediction error as exploration bonus, GCM rewards the states based on the uncertainty to master *skills*. We introduce the concept of *skill* that represents the ability of the agent to achieve a goal  $g$  in a state  $s$ . A skill is mastered when the agent can achieve  $g$  from  $s$  with a high degree of confidence. The confidence is measured in two ways: (1) the difficulty to go from  $s$  to  $g$ , (2) the progress of the agent to achieve this skill. As a result, curiosity incentivizes the agent to seek new knowledge and discover new skills that might lead to the final goal.

A straightforward solution is to consider the final goal as the goal to be achieved in any state. However, in sparse reward environments, reaching the final goal may be infrequent, entailing that for most of the episodes the agent only experiences failures. Therefore, training a probabilistic model for predicting if the final goal is mastered is highly inaccurate. Instead, we propose to estimate if the agent can solve multiple intermediate goals. Since these goals are easier to master—encountered more frequently, the estimation becomes more accurate while carrying information about the final goal. In the absence of domain knowledge, a general choice is to set the goals as states experienced by the agent during exploration.

In details, the goal-based curiosity module takes as input the current observation and produces the curiosity based reward  $r^{gc}$ . The algorithm can be broken down in four parts (Fig. 3). First, at the beginning of each episode,  $K$  goals are sampled with  $1 \leq K \in \mathbb{N}$  (Sect. 4.2.1). Second, at each step, we embed the current observation and the goals into a latent space using a variational autoencoder  $ve : O \rightarrow \mathbb{R}^n$  (Fig. 1). Third, the agent predicts the probability that each goal can be achieved from the current state. Our implementation relies on a deep predictor neural network which predicts the probability that a goal is mastered  $m : \mathbb{R}^n \times \mathbb{R}^n \rightarrow [0,1]$  and the curiosity based reward is calculated. To help the agent in



**Fig. 3** Goal-based curiosity module. The module takes as input an observation  $s$ , and, at the beginning of every episode samples  $K$  goals. The goals and the observation are embedded during step 2,  $\phi(g)$  and  $\phi(s)$  respectively. Step 3 predicts the probability that each goal is mastered; and given this vector of probabilities calculates the curiosity reward signal  $r^{gc}$ . At the end of each episode, multiple new goals are added to the goal buffer based on the states experienced



the quest of new knowledge, we design the reward to encourage the agent to move towards regions of the state-action space that are simultaneously novel and for which the skills are “hard-to-learn”. Namely, we take advantage of uncertainty given the probabilities that the goals are mastered by the agent. We give details about the reward calculation in Sect. 4.2.2. Finally, at the end of the episode, the GCM is updated according to the experienced observations and the predictor neural network is retrained to fit with the new knowledge of the agent (Sect. 4.2.3).

#### 4.2.1 Goals

As described earlier, estimating if the agent can solve the entire task is challenging. Instead, we introduce the concept of *skill* to decompose the problem of reaching a distant final goal into multiple easier goals. A skill is the ability of the agent to reach a goal from a given state. Let  $G$  be the space of possible goals. We suppose that every goal  $g \in G$  corresponds to some predicate  $f_g : S \rightarrow \{0, 1\}$  and that a skill is mastered given a state  $s$  when  $f_g(s) = 1$ . In order to keep a consistent representation, we suppose the goal space  $G$  to be the same as the state space  $S$ ,  $G = \mathbb{R}^n$ .

Working with high-dimensional state spaces such as images is complex. Instead, we embed the goals and states to a latent space  $Z$  using a variational autoencoder  $ve$ ,  $\phi(s) = ve(s)$  and  $\phi(g) = ve(g)$ . In addition to improve data efficiency, it crucially improves generalization capability of our model (Zhao et al. 2018; Rezende et al. 2016). That is, the deep predictor network can generalize knowledge for two goals sharing a similar embedded representation. We initialize the parameters of a VAE by stepping a random agent for a small number of iterations before beginning training. Then, we train a VAE on observations seen along trajectories. The goals are stored in a goal buffer  $R_g$ .

In practice, every episode starts with sampling a sub-set  $g'$  of latent goals given a distribution function  $f_p$ :

$$g' = \{\phi(g) \sim f_p(g)\} \text{ sample } K \text{ goals } \in G \quad (11)$$

In the current implementation, the probability of sampling a goal  $f_p(g)$  is uniform for all the goals. In future work, we anticipate more complex distributions to take into account the difficulty of the goals.

During training, the goals aim to provide additional feedback to the agent to improve exploration. At the end of an episode, we add a mechanism to further enable sample-efficient learning. In addition to the state reached at the end of the episode, we artificially generate new goals by selecting states experienced during the episode. We discuss in Sect. 5.1.4 the strategies to select the new goals after experiencing an epoch. To keep the goals as different as possible, we only add the goals that are different from the existing ones. To do so, we measure the similarity between the embedded goals in a similar manner to that of the RIG (Nair et al. 2018).

$$sim(g1, g2) = -\|\phi(g1) - \phi(g2)\|_A \propto \sqrt{\log ve_\phi(\phi(g2)|g1)} \quad (12)$$

where  $g1$  and  $g2$  are two goals, their embedded representation denoted by  $\phi(g1)$ ,  $\phi(g2)$  respectively, and  $ve_\phi$  the VAE encoder.

#### 4.2.2 Reward calculation

Given the embedding representations of the current state  $\phi(s)$  and the goals, GCM produces the curiosity reward  $r^{gc}$ . We propose to train a predictor network to predict the probabil-

ity that a skill is mastered—the probability that a goal  $\phi(g)$  can be achieved from  $\phi(s)$ ,  $m(\phi(s), \phi(g))$ . The prediction is made for each pair of current state—goal, and the concatenation produces a vector:  $g_{active} = \langle(m(\phi(s), \phi(g_1)), \dots, m(\phi(s), \phi(g_K)))\rangle$ . Given these predictions we define the goal-based curiosity reward:

$$r^{gc} = \left[ \frac{\langle(\alpha) - h\langle(m(\phi(s), \phi(g_1)), \dots, m(\phi(s), \phi(g_K)))\rangle\rangle}{\delta} \right] \tag{13}$$

where  $h$  is a function mapping the probability vector  $g_{active}$  to a real number which expresses the *complexity of the state*. A state is complex when skills are *hard-to-learn*—the goals can be achieved with small probabilities. The parameter  $\delta$  controls the scale of the curiosity reward, and  $\langle\alpha\rangle$  the sign of the intrinsic reward. In the current implementation, we use  $\alpha = \langle 1.0 \rangle$ , a uniform vector of 1, and  $h = \max()$ . The choice of  $h$  depends on the desired exploration behavior. In practice,  $h = \max()$  results in a positive reward unless all skills are considered as mastered by the agent, forcing a complete exploration of the environment. Another choice could be  $h = \text{mean}()$ ; the curiosity signal would decrease when a majority of skills (but not necessarily all) are mastered with high probabilities. As a consequence, the agent is forced to explore novel states, leading to a faster but possibly partial exploration of the state space.

In other words, predicting that the agent cannot achieve a goal results in a positive curiosity-based reward,  $r^{gc} > 0$ . Curiosity pushes the agent to explore novel regions of the state space, and discover more efficient solutions to achieve the goals. One issue with the combination of extrinsic reward and curiosity-based reward is the scaling of the intrinsic reward which may vary between tasks. In order to mitigate this scaling problem, we normalize the curiosity-based reward:

$$r_t^{gc} = \frac{r_t^{gc}}{\sigma(R_{gc})} \tag{14}$$

with  $\sigma(R_{gc})$  an estimation of the standard deviations of the curiosity-based reward returns.

### 4.2.3 GCM update

The VAE network and predictor network are optimized every  $P_{vae}$  and  $P_m$  time-steps respectively. The agent’s observations are collected in a memory bank and every  $P_{vae}$  time-steps the VAE is retrained for 20 epochs. When the memory bank is full, a random element is replaced with the new observation. This is necessary to ensure that the representation of the goals adapts over exploration.

The update of the predictor network can be broken down into three parts: (1) training samples are collected during exploration, (2) a label is assigned to each example, and, (3) the network is optimized. First, the policy interacts with the environment to produce a set of trajectories  $\{\tau^1, \dots, \tau^i\}$ . Within each trajectory, each visited states  $s_t$  is concatenated with the  $d_{max}$  next states  $\{(s_t \cdot s_{t+1}), (s_t \cdot s_{t+2}), \dots, (s_t \cdot s_{t+d_{max}})\}$ . For each training sample  $(s_t, s_{t+y})$ , we consider the left element to be the initial state ( $s$ ), and the right element the goal that the agent aims to achieve ( $g$ ).

Second, a label (mastered/unmastered) is assigned to each example - skill,  $(s, g)$ . To do so, we compute a complexity score for each example  $sc(s, g)$  that takes into account the difficulty to go from  $s$  to  $g$ , and the progress of the agent to achieve this skill. In details, we use a linear combination of the prior knowledge of the predictor network  $p(s, g)$  and the number of time-steps  $dist(s, g)$  between the state  $s$  and the goal  $g$ —skill difficulty:

$$sc(s, g) = \lambda \text{prior}(s, g) + \beta \text{dist}(s, g) \tag{15}$$

where  $prior(s, g)$  is the belief of the predictor network about the skill, and the scalars  $\lambda$  and  $\beta$  weight each component. Given  $sc(s, g)$ , the probability that in a state  $s$  the skill  $(s, g)$  is considered mastered  $p_{skill}(s, g)$  (label +1) follows:

$$p_{skill}(s, g) = 1.0 - \frac{1.0}{1.0 + e^{-b(sc(s,g)-a)}} \quad (16)$$

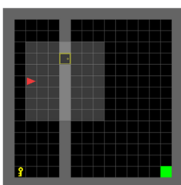
where  $a > 0$  controls the boundary between the two classes. The term  $a$  acts as the degree of certainty to decide whether the agent should explore new parts of the environment or continue to acquire knowledge about this skill. It can be interpreted as follows: to ensure that the skills can be achieved with high certainty,  $a$  should be small. Hence, only low difficulty skills are labeled as mastered (label +1). Thereby, during exploration, GCM produces a positive reward (Eq. 13) in states involving skills that the agent seeks to reinforce (label 0). Finally, the predictor network is optimized every  $P_m = 20000$  time-steps via supervised learning.

## 5 Experiments

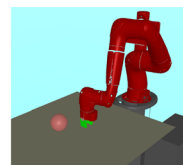
### 5.1 Minigrid

#### 5.1.1 Environment

We evaluate our agent on a set of tasks in the MiniGrid environment (Chevalier-Boisvert et al. 2018). The tasks are partially observable, with varying components such as navigation, sequential solving, and planning. We consider the *Door & key* task domain that we modify to generate RGB images. An example is depicted in Fig. 4a. It consists in a board surrounded by grey walls that the agent cannot cross. The agent is represented by a red triangle, and the final goal to reach by a green square. In order to reach the final goal, the agent has first to pick a key randomly positioned on the map, then unlock a door and finally get to the green square. The agent receives as input a  $32 \times 32$  RGB image of the visible cells, represented by a light gray rectangle. The agent can choose among the 7 possible actions: *turn left*, *turn right*, *move forward*, *pick up an object*, *drop the object being carried*, *open doors - interact with objects* and *complete the task*. Solving such sparse tasks is challenging since the agent only receives a positive reward +1 when it reaches the final goal. Most of the times, the agent only receives 0 as reward. Using efficient exploration strategies such as curiosity is crucial.



(a) Door & key environment (Minigrid)



(b) Visual reacher environment (Mujoco)

**Fig. 4** Examples of training environments. Door & key environment with a board of size  $16 \times 16$  (left). Example of task in the MuJoCo domain consisting in a 7-DoF Sawyer arm that aims to reach a target (right)

### 5.1.2 Implementation details

The agent was trained for 5,000,000 frames. Any RL algorithm could be used to train our model. In our experiments, we combine PPO and GoCu that indicates to perform well. The PPO agent processes the input with a convolutional neural network that consists in a series of three convolution layers, with 16,32,64 filters respectively, kernel size  $2 \times 2$ , stride of size 2 and padding 1, with rectified linear unit (ReLU) function as the activation function. In the experiments, we use a rollout of length 128, an entropy coefficient 0.01, batch size 256, a discount rate  $\lambda = 0.99$ , value loss term coefficient 0.5, and a clipping  $\epsilon = 0.2$ . The experiments were run with 16 parallel environments and, optimized with Adam (Kingma and Ba 2015) for 4 optimization steps and a learning rate  $\alpha = 0.0001$ . At the beginning of an episode,  $K = 50$  goals are sampled from the goal buffer with a maximum size 1500. The intrinsic advantage is combined in PPO with  $\sigma = 0.25$  and  $\delta = 10$  as scaling factor of the curiosity-based reward (Eq. 13). We use  $\lambda = 3$ ,  $\beta = 1$  to weight the prior belief against new knowledge and,  $b = 0.8$ ,  $a = 15$  to assign the label of training samples. In our experiments, we observed that when  $\lambda > 10$ , the agent struggles to integrate new knowledge—discover new skills, whereas a small value (i.e.  $\lambda \leq 2$ ) may result in overfitting of observations. The parameter  $a$  controls the difficulty threshold to classify whether the skill is mastered or not. We found that a value  $10 < a < 30$  performs well in most tasks. Since  $b$  controls the boundary between the two classes, keeping a large value  $b > 0.7$  makes the boundary large, crucial to improving performance of the predictor network.

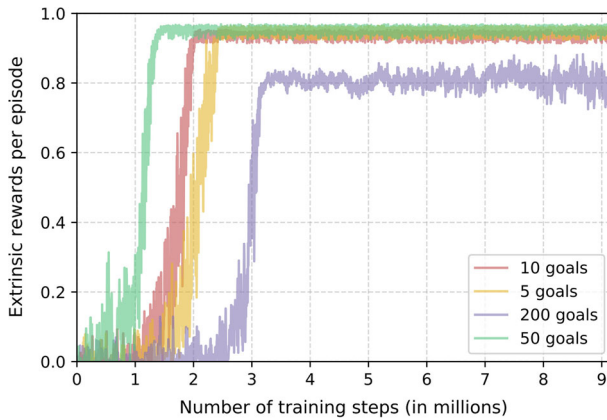
We pretrained the VAE with 10,000 images collected with a random exploration policy. The encoder neural network consists in 3 convolutional layers with the following parameters (filter: 32,32,32 kernel size:  $3 \times 3$ ,  $3 \times 3$ ,  $3 \times 3$ , stride size: 2,1,1) and apply a rectifier non-linearity. They are followed by a fully connected layer of size 128. The latent vector  $\mu$  and  $\sigma$  are of size 10. The 4 last layers are the corresponding decoding layers. We use the compressed representation, the fully connected layer that follows the encoder network, to embed the goals and states. To measure the distance between two goals (Eq. 12), we found that  $A = I$  corresponding to Euclidean distance, performs better than cosine or Mahalanobis distance.

The predictor network is a fully-connected network with 5 hidden layers of size 512. It takes as input the concatenation of the embedding of a state and a goal and outputs (softmax function) the probability that the skill is mastered. The input of each layer is batch normalized and followed by a ReLU activation function. We use Adam to optimize the predictor network, with learning rate 0.0005 on batch of size 128. Every 20,000 training-steps, the network is optimized on the new training samples for 15 epochs.

### 5.1.3 Learning with variable numbers of goals

To isolate how much the number of goals contributes to our algorithm, we vary the number of active goals while fixing other parts of our method. We compare the performance of PPO + GoCu trained with the same hyperparameters and a number of goals  $K \in \{5, 10, 50, 200\}$ . For this experiment, the agents were trained on the *Door & key* task with a board of size  $8 \times 8$ . The goal was to evaluate the gain we can have by increasing the number of goals sampled at each epoch.

The results shown in Fig. 5 show that increasing the number of goals greatly improves performance. However, we also note a too large number of goals,  $K = 200$ , makes the curiosity saturate. Experiments suggest that an excessive number of goals increases prediction errors of the predictor network. In contrast, learning an insufficient number of skills ( $K \in$



**Fig. 5** Extrinsic rewards achieved by PPO + GoCu for different number of goals in the task *Door & key*  $8 \times 8$

{5, 10}) decreases convergence speed. Note that in the  $11 \times 11$  environment the results show similar trends. We observe that  $K = 50$  was optimal, and in practice,  $K \in \{10, 50\}$  works on many tasks.

#### 5.1.4 Goal generation strategies

In this section we experimentally evaluate different strategies for adding new goals to the goal buffer. At the end of an episode,  $l$  new goals are added to the goal buffer, with  $l$  a hyper-parameter that we empirically set to 7. We consider the following strategies to generate the new goals:

- *random- $l$*  states encountered during the last epoch are randomly added to the goal buffer,
- *proportional- $l$*  states are added to the goal buffer with a probability inversely proportional to their order of visit (the final state has the highest probability),
- *late- $l$*  states encountered during the  $l * 4$  last time-steps of the epoch are randomly added to the goal buffer.

All the strategies were tested with the same hyper-parameters. The results comparing these strategies can be found in Table 1. We found that in the task  $5 \times 5$ , all the strategies lead to similar performance. As can also be seen, the *proportional* strategy is the most efficient one in term of convergence speed and average reward (at convergence) in the task  $8 \times 8$  and  $11 \times 11$ . The *proportional* strategy achieves an almost perfect score in the three tasks.

#### 5.1.5 Overall performance

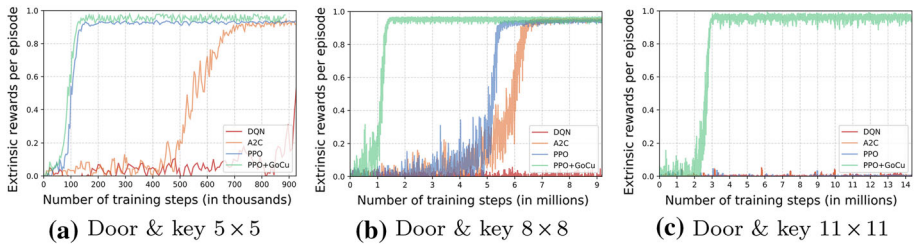
In this section, we perform a set of two experiments to evaluate the overall performance of our algorithm. First, we verify if GoCu achieves better performance than traditional RL methods. Second, we compare GoCu against state-of-the-art curiosity-based learners.

In order to verify that GoCu improves performance, we evaluate PPO (Schulman et al. 2017) with and without GoCu on the *Door & key* task. Moreover, we compare against A2C (Mnih et al. 2016) and DQN (Mnih et al. 2013, 2015). We report the average extrinsic reward supplied by the environment. We present in Fig. 6 the evolution of the extrinsic

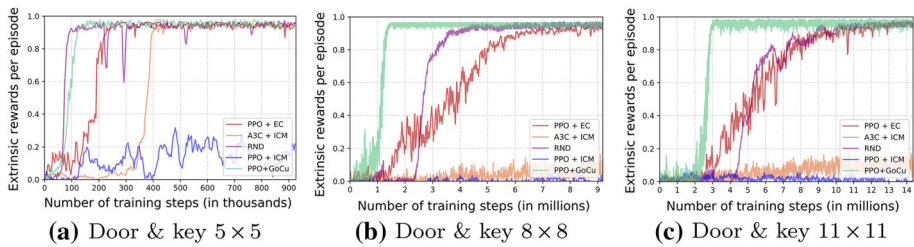
**Table 1** The table compares the average reward for various goal generation strategies, and on different tasks

Method	Time-steps to converge (in millions)			Average reward (at convergence)		
	5 × 5	8 × 8	11 × 11	5 × 5	8 × 8	11 × 11
Random	<b>0.22 ± 0.05</b>	2.20 ± 0.10	4.1 ± 0.42	0.96 ± 0.04	0.94 ± 0.05	0.94 ± 0.07
Proportional	0.25 ± 0.01	<b>1.60 ± 0.04</b>	<b>3.3 ± 0.16</b>	<b>0.96 ± 0.03</b>	<b>0.95 ± 0.05</b>	<b>0.95 ± 0.08</b>
Late	0.32 ± 0.03	2.56 ± 0.21	3.75 ± 0.36	0.94 ± 0.04	0.95 ± 0.06	0.94 ± 0.10

Performance are averaged over 5 runs using PPO + GoCu as learning method  
 Bold values indicate the best performing method



**Fig. 6** Extrinsic rewards achieved by the agent on the *Door & key* task. An episode is considered successful when the agent reaches the final goal resulting in a +1 reward



**Fig. 7** Comparison of GoCu algorithm (green) to other algorithms on several environments with varying size (degree of sparsity) (Color figure online)

reward achieved by the agent. In such sparse tasks, the DQN agent fails to solve the task. The learning curve shows that our agent outperforms PPO trained without GoCu as well as A2C. After converging, PPO + GoCu achieves an almost perfect score in 95% of the runs, slightly higher than PPO trained without curiosity, 93%. Only our method scales with the size of the environment and is significantly faster in term of convergence speed.

In addition, we also conduct a set of experiments to compare GoCu with RND (Burda et al. 2018), PPO + EC (Savinov et al. 2019), A3C + ICM (Pathak et al. 2017), and, PPO + ICM (Pathak et al. 2017). We perform an evaluation using the same open-source implementation with exactly the same hyper-parameters of the original works. We ran the agents for 3 runs each for 1 million time-steps on the *Door & key*  $5 \times 5$  task, 10 millions time-steps on *Door & key*  $8 \times 8$ , and, 15 millions time-steps for the board of size  $11 \times 11$ . We can see in Fig. 7 that in 2 out 3 domains our method is significantly faster than the baselines and improves the performance of PPO, while on task  $5 \times 5$ , we obtain similar performance with RND. We also observe that PPO + ICM and A3C + ICM result in poor performance in the early stages of learning in the three tasks and was not able to reach the goal. Note that we didn't fine-tune hyper-parameters for each domain, that might be an opportunity to further improve performance.

## 5.2 Simulated robotics

### 5.2.1 Environment

Our second set of experiments explores domains from the Mujoco control benchmark (Todorov et al. 2012). We evaluate performance on five tasks. (1) *Visual Reacher*: a 7-DoF Sawyer arm learns to reach goal positions. The end-effector (EE) is constrained to a 2-dimensional rectangle. (2) *Visual Door Hook*: a 7-DoF Sawyer arm with a hook on the end



of the effector to handle a door hook. (3) *Visual Pusher* and (4) *Visual Multi-Object pusher*: the arm aims to push one or two pucks to a target. Note that the end-effector is constrained to only move in the XY plane. (5) *Visual Pick and Place*: a Sawyer arm with a grab at the end of the effector to grab a small ball that the agent has to place at a target. We extend the tasks to produce  $32 \times 32$  RGB observations. The rewards are sparse indicator functions being the distance in the final state between the objects and their goals. We show an example of two tasks in Fig. 4b.

### 5.2.2 Implementation details

For these tasks, we keep the hyper-parameters of PPO unchanged for both augmented and baseline agents. We use Adam to optimize the predictor network as well as PPO. For these tasks, we pretrained the VAE with 10,000 images. We compare our method with the following prior works: A3C + ICM (Pathak et al. 2017), PPO + ICM (Pathak et al. 2017), PPO + EC (Savinov et al. 2019), HER (Hindsight Experience Replay) (Andrychowicz et al. 2017), RIG (RL with Imagined Goals) (Nair et al. 2018) and, DSAE (Deep Spatial Autoencoders) (Finn et al. 2016).

The parameters of GoCu are similar as for the minigrid domain. For this range of tasks, the predictor network consists in a fully-connected network with 3 hidden layers of size 512. Adam optimizer was used with learning rate 0.0003 and batch size 64. We decrease the number of active goals to 25 and the capacity of the goal buffer to 1000.

### 5.2.3 Overall performance

Agents were trained for 500K time steps. We trained 5 agents separately without additional parameters tuning. Table 2 reports the final distance to the goal of our method against six baselines. In these experiments, we observe that HER tends to perform poorly due to the visual representation of the tasks. RIG achieves higher performance in the visual pick and place task thanks to the use of sub-goals that guide the agent towards the goal. From Table 2, it is clear that PPO + ICM is significantly outperformed, we suspect that the high similarity among states results in small rewards. On the other hand, our curiosity-based approach outperforms the prior methods only using visual information in 3 out of 5 domains. Overall, the results show that our method can learn policies in the robotic domain. It confirms that GoCu is a crucial element which makes learning from sparse rewards possible.

## 5.3 Atari games

We also evaluate the proposed curiosity method on three difficult exploration Atari 2600 games from the Arcade Learning Environment (ALE) (Abbeel et al. 2007): Montezuma's Revenge, Private Eye, and, Pitfall. In the selected games, training an agent with a poor exploration strategy often results in a suboptimal policy. We first compare to the performance of a baseline A2C, and, PPO implementation without intrinsic reward. The results are shown in Table 3. In these games, training an agent with a poor exploration strategy results in a score close to zero, except for Montezuma's Revenge that PPO could partially solve.

In addition, we also compare our method against other techniques using alternative exploration bonus (Table 3). In Pitfall, PPO + GoCu achieves similar performance as RND. Many interactions yield negative rewards that dissuade our agent and other baselines from exploring efficiently the environment. Despite this issue, PPO + GoCu performs better than algorithms



**Table 2** Comparison of seven models over five tasks in the Mujoco domain

Method	Final distance goal (at convergence)				
	Visual reacher	Visual door hook	Visual pusher	Visual multi-object pusher	Visual pick and place
HER	0.09 ± 0.01	0.42 ± 0.02	0.26 ± 0.02	0.36 ± 0.03	0.19 ± 0.01
A3C + ICM	0.15 ± 0.03	0.41 ± 0.04	0.32 ± 0.04	0.35 ± 0.03	0.17 ± 0.02
PPO + ICM	0.08 ± 0.01	0.32 ± 0.02	0.11 ± 0.01	0.21 ± 0.02	0.12 ± 0.01
PPO + EC	0.08 ± 0.02	0.11 ± 0.01	0.12 ± 0.02	0.18 ± 0.03	0.11 ± 0.01
RIG	<b>0.07 ± 0.02</b>	0.08 ± 0.01	0.17 ± 0.02	0.10 ± 0.02	<b>0.08 ± 0.01</b>
DSAE	0.10 ± 0.01	0.28 ± 0.05	0.25 ± 0.01	0.27 ± 0.02	0.13 ± 0.01
PPO + GoCu	<b>0.07 ± 0.01</b>	<b>0.06 ± 0.02</b>	<b>0.09 ± 0.02</b>	<b>0.06 ± 0.03</b>	0.13 ± 0.02

We report the final distance to goal for 5 runs that were performed for 500K simulations steps

Bold values indicate the best performing method

**Table 3** Final mean score of our method and baselines on Atari games

Method	Maximum Mean Score (at convergence)		
	Montezuma's revenge	Private eye	Pitfall
A2C (Mnih et al. 2016)	13	574	− 15
PPO (Schulman et al. 2017)	2497	105	− 32
RND (Burda et al. 2018)	8152	8666	− 3
PPO + EC (Savinov et al. 2019)	8032	9258	− 10
PPO + ICM (Pathak et al. 2017)	351	503	− 14
DeepCS (Stanton and Clune 2019)	−	1105	− 186
Average Human (Wang et al. 2016)	4753	69,571	6464
PPO + GoCu (Ours)	<b>10,958</b>	<b>12,546</b>	− 4

We report the results achieved over total 100M timesteps of training, averaged over 3 seeds  
 Bold values indicate the best performing method

without curiosity. However, in Montezuma's Revenge, PPO + GoCu outperforms average human performance and previous state of the art. On Private PPO + GoCu's performance are higher than other baselines. The strong performance of PPO + GoCu on these tasks suggests that encouraging the agent to learn new skills is a new mean to enable learning in sparse-reward domains.

## 6 Conclusion

In this work we introduce a new mechanism for generating curiosity, easy to implement, and that scales to high dimensional inputs such as images. It enables an arbitrary RL agent to acquire new skills using self-generated curiosity. A key concept is to reward the states with associated skills difficult to master; to move towards regions of the state-action space that are "hard-to-learn". Furthermore, by representing the states and goals with a VAE, we can improve data efficiency as well as improving generalization. We have shown its ability to solve complex tasks with continuous state spaces, and exceeds baseline agents in term of overall performance and convergence speed. To support these claims, we presented an evaluation on a set complex and sparse visual tasks from Minigrid, MuJoCo, and Atari games. An interesting future research direction is to share the goals among tasks to improve generalization and reduce training time when the task becomes more complex. For example in Minigrid, the goal *collect the key* is shared among the tasks, independently of the size of the environment.

**Acknowledgements** This work was supported in part by SOKENDAI.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

Abbeel, P., Coates, A., Quigley, M., & Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of advances in neural information processing systems* (pp. 1–8).

- Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., & Welinder, P. et al. (2017). Hindsight experience replay. In *Proceedings of advances in neural information processing systems*.
- Baranes, A., & Oudeyer, P. Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1), 49–73.
- Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Proceedings of advances in neural information processing systems* (pp. 1471–1479).
- Burda, Y., Edwards, H., Storkey, A. J., & Klimov, O. (2018). Exploration by random network distillation. [arXiv:1810.12894](https://arxiv.org/abs/1810.12894).
- Chevalier-Boisvert, M., Willems, L., & Pal, S. (2018). Minimalistic gridworld environment for openai gym. <https://github.com/maximecb/gym-minigrid>.
- Eppe, M., Magg, S., & Wermter, S. (2018). Curriculum goal masking for continuous deep reinforcement learning. CoRR. [arXiv:1809.06146](https://arxiv.org/abs/1809.06146).
- Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., & Abbeel, P. (2016). Deep spatial autoencoders for visuomotor learning. In *Proceedings of the international conference on robotics and automation, IEEE* (pp. 512–519).
- Florensa, C., Held, D., Wulfmeier, M., Zhang, M., & Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. [arXiv preprint arXiv:1707.05300](https://arxiv.org/abs/1707.05300).
- Held, D., Geng, X., Florensa, C., & Abbeel, P. (2017). Automatic goal generation for reinforcement learning agents. [arXiv preprint arXiv:1705.06366](https://arxiv.org/abs/1705.06366).
- Itti, L., & Baldi, P. F. (2006). Bayesian surprise attracts human attention. In *Proceedings of advances in neural information processing systems* (pp. 547–554).
- Jaderberg, M., Mnih, V., Czarniecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. In *International conference on learning representations*.
- Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. In *Proceedings of the international conference on learning representations*.
- Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *Proceedings of the international conference on learning representations*.
- Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).
- Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2), 189–223.
- Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1), 1334–1373.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., & Wierstra, D. et al. (2013). Playing atari with deep reinforcement learning. [arXiv preprint arXiv:1312.5602](https://arxiv.org/abs/1312.5602).
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., & Harley, T. et al. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the international conference on machine learning*.
- Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., & Levine, S. (2018). Visual reinforcement learning with imagined goals. In *Proceedings of advances in neural information processing systems* (pp. 9209–9220).
- Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the seventeen international conference on machine learning, Citeseer*.
- Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the International Conference on Machine Learning*, 99, 278–287.
- Ostrovski, G., Bellemare, M. G., van den Oord, A., & Munos, R. (2017). Count-based exploration with neural density models. In *Proceedings of the international conference on machine learning* (pp. 2721–2730).
- Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the international conference on international conference on machine learning*.
- Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., & Jimenez Rezende, D. et al. (2017). Imagination-augmented agents for deep reinforcement learning. In *Proceedings of advances in neural information processing systems*.
- Rezende, D.J., Mohamed, S., Danihelka, I., Gregor, K., & Wierstra, D. (2016). One-shot generalization in deep generative models. In *Proceedings of the international conference on international conference on machine learning* (pp. 1521–1529).

- Salge, C., Glackin, C., & Polani, D. (2014). Changing the environment based on empowerment as intrinsic motivation. *Entropy*, *16*(5), 2789–2819.
- Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., & Gelly, S. (2019). Episodic curiosity through reachability. In *Proceedings of the international conference on learning representations*.
- Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. In *Proceedings of the international conference on machine learning*.
- Schmidhuber, J. (1991a). Curious model-building control systems. In *Proceedings of the IEEE international joint conference on neural networks, IEEE*.
- Schmidhuber, J. (1991b). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the international conference on simulation of adaptive behavior: From animals to animats* (pp. 222–227).
- Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., & Moritz, P. (2015). Trust region policy optimization. *Proceedings of the International Conference on International Conference on Machine Learning*, *37*, 1889–1897.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347).
- Stadie, B. C., Levine, S., & Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. arXiv preprint [arXiv:1507.00814](https://arxiv.org/abs/1507.00814).
- Stanton, C., & Clune, J. (2019). Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. In *Proceedings of the international conference on international conference on machine learning*.
- Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT Press.
- Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (pp. 1057–1063).
- Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *Proceedings of the international conference on intelligent robots and systems* (pp. 5026–5033).
- Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the international conference on international conference on machine learning*.
- Zhao, S., Ren, H., Yuan, A., Song, J., Goodman, N., & Ermon, S. (2018). Bias and generalization in deep generative models: An empirical study. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds). *Proceedings of advances in neural information processing systems* (pp. 10792–10801).

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.