Review on Curiosity Driven Reinforcement Learning

Lucas Gerretsen
Ecole Normale Supérieure de Paris-Saclay
94230 Cachan
lucas.gerretsen2@gmail.com

Alexandre Zouaoui
Ecole Normale Supérieure de Paris-Saclay
94230 Cachan
alexandre.zouaoui@telecom-paristech.fr

Abstract

In Reinforcement Learning, training an agent in a supervised setting of gathering high extrinsic rewards from an environment is still nowadays an unsolved problem. This is especially difficult for large realistic environments where observations are partial or where high rewards are sparse. For the latter case, many methods propose a model of intrinsic rewards that behave as curiosity, in order to guide agents in their exploration to find novel states. As opposed to count-based methods, some recent ones have modeled curiosity as an error of prediction of observed states and have shown state-of-the-art performances on artificial toy environments. The goal of this work is to review some of these methods.

1 Introduction

Reinforcement Learning (RL) lies in sequential settings of interactions between an agent and an environment, where the agent dynamically learns a behaviour (i.e. a policy) to maximize rewards provided by an environment. This learning requires the agent to randomly explore new states to try and come across interesting high rewards, and then to understand which action lead to this reward. In deterministic environments, naively memorizing the sequence of actions to reproduce is theoretically enough. This is however not robust to stochastic environments where one sequence of actions may not lead to one deterministic state. This is why RL agents rather learn a policy function that transforms an observed state into an action, and combine this policy with some sort of exploration to enforce the agent to perform exploration along with exploitation.

Learning such policy first require a numeric representation of the observed state. RL breakthroughs such as learning to play Backgammon [15] or Go [10] used state representation hand-crafted by experts in these games, to help the agent in learning a policy with a-priori meaningful information as inputs. Lately, the Arcade Learning Environment (ALE, also referred to as Atari 2600 in the literature) [13] was proposed as a set of 49 environments to perform comparable experiments in RL, with no hand-crafted representation. There, observed states are images which may be lightly preprocessed with the same function across the 49 environments: [11] rescaled input images by downsampling to reduce input data, and maxed pairs of images to facilitate understanding of 2-frames flickering.

Authors of ALE also performed baseline benchmarks with the SARSA algorithm. The DeepQ approach proposed by [11] to solve these environments was a breakthrough, beating SARSA on 43 out of 49 environments. This approach, although it used deep neural networks to approximate Q functions, was quite simple considering that the exploration is ensured by an ϵ-greedy policy, which is considered as the baseline method for performing exploration in RL according to [14].

Moreover, some RL environments are said to have sparse rewards when the set of possible states is much larger that the set of states which contain a non-zero reward. Such environments require more efficient exploration than random walks generated by ϵ-greedy policies. Similarly to how living beings learn by being curious, various approaches propose efficient models of intrinsic rewards, as opposed to relying solely on the environment extrinsic rewards. These environments with

sparse rewards are quite realistic in that sense; moreover, adding hand-crafted extrinsic rewards to an environment is humanly time-consuming or may even induce bias. Intrinsic rewards are a way of enforcing the agent to explore states that were so far too rarely visited as compared to others. We can distinguish count-based exploration [6], from curiosity-driven exploration [8] [1] [4] where curiosity is considered an error of prediction of the environment's dynamics.

Montezuma's revenge, one of the hardest environments from ALE, still remains a challenge as shown in 1. Its difficulty is probably due to the joint complexity of exploring new states while learning numerous skills for achieving them. While [13] and [11] achieved scores no better than 0, curiosity-driven algorithms [2] and [1] respectively achieved state-of-the-art scores of 2505 and 11347. One different approach from [3] did achieve 35410, beating the average human expert's 34900, but they used state resets, therefore benefiting from the Atari emulator's properties and the environment being deterministic, and also used imitation learning algorithms to learn from human experts demonstration.

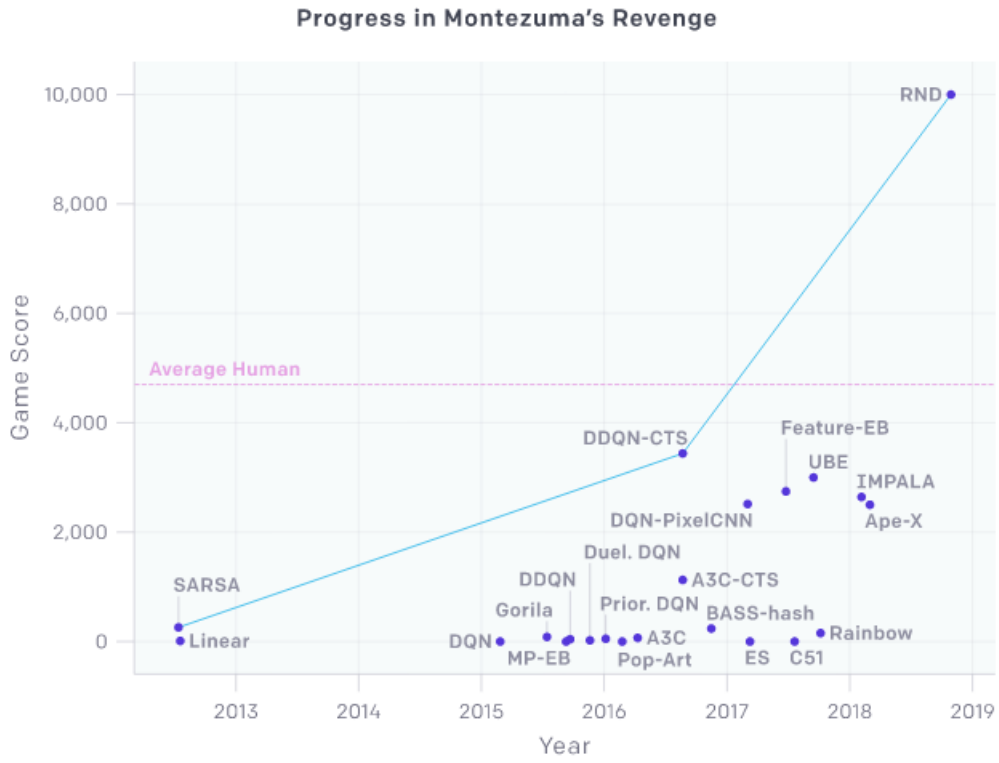


Figure 1: State-of-the-art algorithms on Montezuma's Revenge over time. This figure does not include [3] as Go-Explore uses imitation learning and state resets where [1] does not.
Source: https://blog.openai.com/reinforcement-learning-with-prediction-based-rewards/ (blog post from [1])

The latter experiments rise the question of robustness of RL algorithms to stochastic environments. Environments from ALE are all deterministic, and randomness in the benchmarks are generally achieved by simply waiting variable amounts of frames at the start of each episode (from 0 to 30 frames in [11]) to avoid overfitting.

Contributions

The contributions of this review paper are three-fold. We first review the Proximal Policy Optimization (PPO) algorithms family as they have been successful in outperforming other online policy gradient methods [5]. We then present the Intrinsic Curiosity Module (ICM) that aims to simulate

the agent curiosity to explore novel states [4]. Lastly, we focus on an exploration bonus introduced for deep reinforcement learning methods using Random Network Distillation (RND) [1].

2 Presentation of the methods

2.1 On Proximal Policy Optimization (PPO)

We have seen during the class how to solve exactly a Markov Decision Process (MDP). However it requires to have an exact definition of the transition probabilities $p(.|x,a)$ and the reward function $r(x,a)$ and this is often not the case. Similarly, Dynamic programming requires exact representations of the value functions and policies but this is typically not tractable in a continuous state space as it is often the case in RL applications. There is therefore a need for approximations. Policy Gradient methods take place in these settings as they work by computing an estimator of the policy gradient to be used in a stochastic gradient ascent algorithm. We have seen during the course that a common gradient estimator is

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta \log_{\pi_\theta}(a_t|s_t)\hat{A}_t] \tag{1}$$

where π_θ is a parametrized stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep t ($A_t = Q^{\pi_\theta}(x_t,a_t) - V^{\pi_\theta}(x_t)$). The notation $\hat{\mathbb{E}}_t$ hints that we are dealing with an empirical average over a batch of samples. Policy gradient methods alternate between collecting samples and optimizing the policy over the collected episodes in an online fashion.

Caution must be taken when attempting to optimize the loss function associated with $\hat{g}$ by performing multiple steps of gradient ascent as it may lead to taking huge steps in the policy update. We have seen how conservation policy iteration algorithms such as Trusted Region Policy Optimization (TRPO) [12] can alleviate this issue by introducing a constraint on the size of the policy update using Kullback-Leibner divergence:

$$\text{maximize}_\theta \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t] \quad \text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(.|s_t), \pi_\theta(.|s_t)]] \leq \delta \tag{2}$$

The authors of PPO [5] aim to introduce an algorithm that exhibits similar data efficiency and performance as TRPO [12] using only first-order optimization and therefore scale better. They propose a novel surrogate loss function that has clipped probability ratios. Their approach derives another pessimistic formulation of the optimization problem we are trying to solve. In essence, their policy optimization scheme requires to alternate between sampling data from the current step policy and performing several update steps on the sampled data.

Let $r_t(\theta)$ denote the aforementioned probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\,\text{old}}(a_t|s_t)}$. Using this notation, TRPO maximizes the following surrogate objective function under a constraint on the Kullback-Leibner divergence on the two successive policies:

$$L_{TRPO}(\theta) = \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t] = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t] \tag{3}$$

The authors of PPO propose to modify this surrogate objective by introducing a clip on the probability ratio as follow:

$$L_{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{4}$$

where ϵ is a hyperparameter to be set empirically. Using this formulation, the surrogate objective is a lower bound on the unclipped objective from 3 that tends to move to far away from the current policy. In an automatic differentiation framework, L_{CLIP} replaces the commonly used loss function associated with 1.

In the policy gradient implementation proposed in [9] that is well-suited to using recurrent neural networks, the policy is run for T timesteps and is updated using the samples collected. This approach requires an advantage function estimator that does not look beyond T steps, such as:

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \ldots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{5}$$

where $t \in [O,T]$. Here r_t are the rewards obtain along the trajectory and are not to be confused with the probability ratio introduced above.

In that setting we can derive the Actor-Critic-Style PPO algorithm 1 that alternates between sampling data from the current policy (which can be done in parallel) and updating the policy by performing several update steps using the clipped surrogate loss 4 with respect to the policy parameter θ on the collected samples.

Algorithm 1 PPO (Actor-Critic Style)

for iteration = 1, 2, ... **do**
  for actor = 1, ..., N **do** ▷ This enables parallel exploration
    Run policy π_{old} in environment for T timesteps
    Compute advantage function estimates $\hat{A}_1, \ldots, \hat{A}_T$
  Optimize surrogate loss L_{CLIP} w.r.t. θ using K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$

The authors then perform an extensive experiment to determine the clipping hyperparameter ϵ, using 7 different environments, each with 3 random seeds. Based on their experimental setup, they concluded that $\epsilon = 0.2$.

In the following sections we will see how PPO can be used in a deep reinforcement learning setting.

2.2 On the Insintric Curiosity Module (ICM)

In the context of building dynamic intrinsic rewards described in 1, [4] recently pointed out some desirable properties for an artificially generated curiosity. As they stated it, an agent interacting with an environment should become curious about specific observable changes in the environment, and not for instance get distracted by irrelevant observations such as natural movements of trees.

Having this in mind, they categorized such information about the observations into 3 disjoint sets:

- (C_1) things that the agent can control,
- (C_2) things that the agent cannot control but that can affect it,
- (C_3) the rest.

They then described their Intrinsic Curiosity Module (ICM) to be added on top of Asynchronous Advantage Actor-Critic (A3C) [9]. Note that A3C in A3C+ICM was later replaced by Proximal Policy Optimization (PPO) [5] in [2]. They stated that A3C+ICM trains agents that follow property P_1: "the agent learns a representation function ϕ of the observed states that filters out any information from set C_3". Their model is close to the one proposed by [7], but uses the error of the forward model to provide intrinsic rewards.

A3C is the top-level part of the agent which observes states s_t and rewards $r_t^e + r_t^i$, computes policies π_t, and outputs actions a_t. States and extrinsic rewards are provided by the environment itself and the intrinsic reward is provided by ICM. ICM is essentially made of 3 components shown in Fig 2: a state representation model ϕ, forward model f, and an inverse dynamics model g_{idm}. This module follows 2 goals: it provides dynamic intrinsic rewards r_t^i at each transition (s_t, a_t, s_{t+1}), and it learns ϕ such that the agent follows property P_1. In particular, parameters from the representation model ϕ may be shared with some parameters of A3C.

Similarly to RND (see 2.3), the forward model $\widehat{\phi(s_{t+1})} = f(a_t, \phi(s_t); \theta_f)$ is used to predict an expected state representation. The prediction error $||\widehat{\phi(s_{t+1})} - \phi(s_{t+1})||_2^2$ is both used as an intrinsic reward, meaning that high errors correspond to rarely-seen-before states, and optimized such that future predictions should be more accurate to result in lower intrinsic rewards. Therefore, the agent should obtain decreasing intrinsic rewards from states it observes on a regular basis.

The inverse dynamics model takes representations $\phi(s_t)$ and $\phi(s_{t+1})$ as inputs and outputs $\widehat{\pi} = g_{idm}(\phi(s_t), \phi(s_{t+1}); \theta_g, \theta_\phi)$. It can be seen as predicting action $\widehat{a_t} = \arg\max \widehat{\pi}_t$ that lead from state s_t to s_{t+1}. In the discrete case, the prediction error is computed as a cross-entropy between the predicted $\widehat{\pi}_t$ and the one hot real action a_t, which can be rewritten $log(\widehat{\pi}_t(a_t))$. Note that it is the optimization of g_{idm} that affects learning of ϕ, which supports property P_1: indeed, learning ϕ by optimizing g_{idm} should not be affected by observations that happen independently of taken actions.

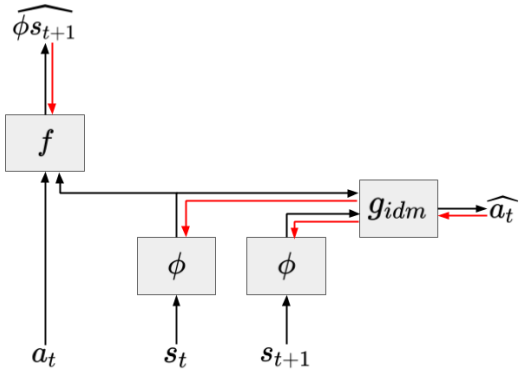
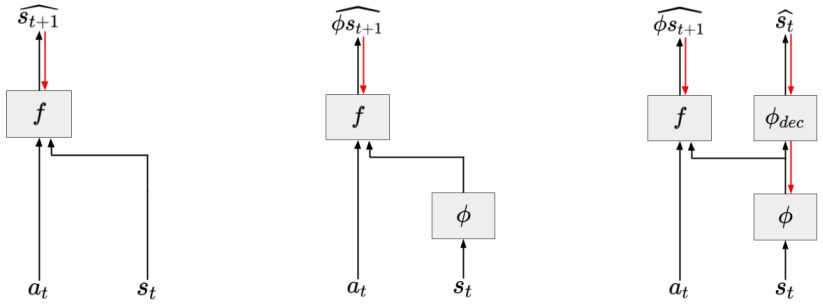


Figure 2: Functional graph of ICM. At each transition, ICM is given a transition (s_t, a_t, s_{t_+1}) and outputs an intrinsic reward proportional to the error on the output of f. Red arrows indicate gradient backpropagations.

To help understanding pros and cons of ICM (ie ICM+IDM), [2] briefly presented 3 alternative architectures for ICM: ICM+pixels, ICM+RF (Random Features) and ICM+VAE (Variational Auto-Encoder), whose architectures are shown in 3; they described how they each have pros and cons. RF, VAE and IDM provide notably smaller feature representation than raw pixels which helps the forward model to better learn states that were visited, and to outptut more accurate intrinsic rewards. RF may theoretically abstract the observations too much and lose crucial information, while on the other hand VAE should converge to sufficient feature representation. Though, unlike IDM, VAE may represent too much information and thus not follow P_1. Considering this reasoning, we could see the problem of choosing the right feature representation to follow P_1 as a compromise between representation efficiencies of RF and VAE. Moreover, finding such feature representation necessarily requires learning, which causes the representations provided to the forward model f to be not stationary; in other words, while learning of the feature representation, curiosity may become inconsistent.



(a) ICM+pixels (b) ICM+RF (c) ICM+VAE

Figure 3: 3 alternative versions of ICM experimented by [2] against the original ICM+IDF. Red arrows indicate gradient backpropagations.

Also, experiments by [2] suggest that this compromise is even harder to obtain. Note that, since intrinsic rewards provided by curiosity models are essentially unsupervised rewards, they may not align with extrinsic rewards that are task-dependent. In a supervised setup where we are given extrinsic rewards, we could modify definitions of above sets in the following way:

- (C'_1) things that the agent can control and that can affect its reaching extrinsic rewards,
- (C'_2) things that the agent cannot control but that can affect its reaching extrinsic rewards,
- (C'_3) the rest.

We now define property P_2 similarly to P_1, but using set C'_3. This property is more desirable for an agent than P_1; also, given that, by definition, $C_3 \subset C'_3$, we have that $P_2 \Rightarrow P_1$ but $P_1 \nRightarrow P_2$. These

are exactly shown in experiments by [2] on ICM+A3C where they added a noisy television screen on a wall in a maze: ICM-trained agents were not affected by this screen, which confirms that they follow P_1, except when they were given a remote to control the screen, which proves that they do not follow P_2. Even though they could achieve the supervised task, they needed many iterations for this which, as they mentioned, asks questions about robustness of algorithms to stochastic environments.

In practice, in these experiments, ICM+RF performed worse than ICM+IDM but both performances are still relatively close in the maze environment as compared to vanilla PPO by [5] which suggests that even random features could generalize well to other environments.

2.3 On Random Network Distillation (RND)

In this subsection we present the Random Network Distillation bonus exploration introduced in [1]. The authors are building on recent advances in curiosity-based exploration using intrinsic reward to make up for a lack of meaningful rewards coming directly from the environment. In particular, the Random Network Distillation can be seen as a refinement of the ICM [4] presented above. Their main achievement consists in making huge progress on solving the Montezuma's Revenge Atari game present in the ALE.

On properties

The exploration bonus introduced in [1] has favorable properties as it is able to work well with high-dimensional observations such as the images of a game and is efficient since it only requires a single forward pass on a neural network on a batch of collected samples.

The intuition behind the exploration bonus they present lies in the observation that neural networks tend to have lower prediction errors on examples that are similar to those they have been trained on. As a result, the authors propose using the prediction errors of a network trained on the agent's past experiences to assess the novelty of new experiences.

On Random Features

To counter-act the noisy TV problem alluded to in 2.2, where agents that maximize a prediction error are prone to get attracted by transitions where the answer to the prediction problem is a stochastic function of the input, the authors define an exploration bonus using a prediction problem where the answer is a deterministic function of the input. Here the prediction problems boils down to guessing the output of a fixed and randomly initialized neural network on the current observation.

As in 2.2, we denote the reward r_t as the sum of the extrinsic reward r_t^e and the intrinsic reward r_t^i. The fixed and randomly initialized target neural network encodes an observation into an embedding using $f : \mathcal{O} \mapsto \mathbb{R}^k$ while the predictor neural network aims to approximate f by $\hat{f} : \mathcal{O} \mapsto \mathbb{R}^k$. The trick is to train the predictor network to minimize the expected MSE between $\hat{f}(.;\theta)$ and f with respect to its parameters $\theta_{\hat{f}}$.

The authors then attribute the prediction errors to a few factors:

1. **Factor 1:** When the model fails to generalize from previously seen examples. Novel experience then corresponds to high prediction error.
2. **Factor 2:** The prediction error is high due to the prediction target being stochastic.
3. **Factor 3:** The prediction error stems from lack of information necessary for making a good prediction, or the model class of the predictor does not fit the complexity of the target, using neat error bars in the proccess.

While Factor 1 is exactly what we are looking for to model the intrinsic reward r_t^i, Factors 2 and 3 are not desirable. The beauty of the RND exploration bonus is that it precisely addresses these factors, insofar as a fixed randomly initialized neural network will always yield a deterministic embedding given the same observation as input. Moreover, the target and predictor models having the same architecture, they exhibit the same complexity.

On PPO

One of the challenges raised by adding the intrinsic reward signal r_t^i is that intrinsic and extrinsic rewards should not be treated in the same way. Intuitively, we want intrinsic rewards to accumulate over an infinite horizon whereas extrinsic rewards should only accumulate during the current episode. This requires combining episodic and non-episodic returns and tweaking the discount of each reward type. The authors do so by revisiting the PPO algorithm explained in 2.1 and introduce two value heads for the two reward streams. They believe this added flexibility was key in achieving their above average human performance on Montezuma's Revenge.

The authors conduct exhaustive experiments to set the discount rates γ_i and γ_e. Furthermore, they motivate their choice for Recurrent Neural Networks over Convolutional Neural Networks based on their results.

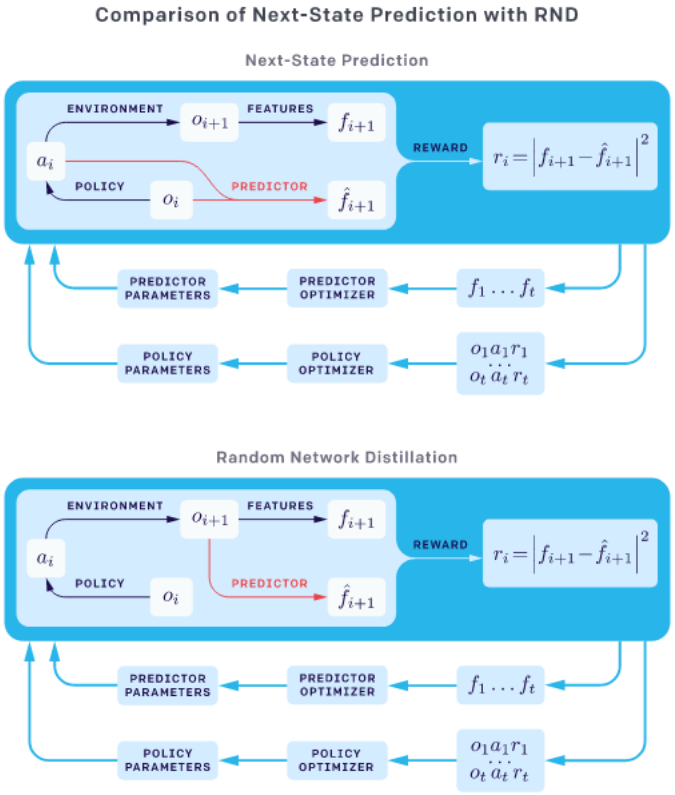


Figure 4:
Source: https://blog.openai.com/reinforcement-learning-with-prediction-based-rewards/ (blog post from [1])

Figure 4 highlights the key difference between a next-state prediction approach as opposed to predicting the output of a randomly initialized network given the next state observation (RND).

3 Conclusion and perspective

Summary of the results obtained

In this review we have investigated curiosity-based approaches to solving RL problems where the environment only delivers a sparse reward signal. We mentioned the PPO [5] as an efficient policy gradient method, that has been revisited in the RND [1] exploration bonus implementation to tackle the hallmark of Atari Learning Environment games, that is Montezuma's Revenge, and in [2]. In addition we reviewed the desirable properties an Intrinsic Curiosity Module [**ICM**] should exhibit to successfully generate an artificial curiosity to learning agents.

It should be noted that the approach described in [1] only tackles the local exploration problem (i.e. the short-term decisions) but is not sufficient for global exploration that would require coordinated decisions over longer time horizons.

We would like to thank the authors of [4], [5] and [1] for the clarity of their paper as well as their thoughtfully designed experiments. It is clear they are striving to lead RL field to be more accountable when it comes to reproducibility, which we agree with, as they carefully detail how they pick hyperparameters and their design choices.

Possible improvement/extension

We plan on conducting a few experiments on well chosen environments from the OpenAI Gym, to be presented during the final presentation. Such experiments could also help envision limits of the described algorithms when confronted to stochastic setups.

References

[1] Yuri Burda et al. “Exploration by Random Network Distillation”. In: *CoRR* (2018).

[2] Yuri Burda et al. “Large-Scale Study of Curiosity-Driven Learning”. In: *CoRR* (2018).

[3] Adrien Ecoffet et al. *Montezuma’s Revenge Solved by Go-Explore, a New Algorithm for Hard-Exploration Problems (Sets Records on Pitfall, Too)*. 2018.

[4] Deepak Pathak et al. “Curiosity-driven Exploration by Self-supervised Prediction”. In: *CoRR* (2017).

[5] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *CoRR* (2017).

[6] Haoran Tang et al. “#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 2753–2762.

[7] Pulkit Agrawal et al. “Learning to poke by poking: Experiential learning of intuitive physics”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 5074–5082.

[8] Rein Houthooft et al. “VIME: Variational Information Maximizing Exploration”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 1109–1117.

[9] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. 2016, pp. 1928–1937.

[10] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 7587 (2016), p. 484.

[11] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 7540 (2015), p. 529.

[12] John Schulman et al. “Trust Region Policy Optimization”. In: *ICML*. 2015.

[13] Marc G Bellemare et al. “The arcade learning environment: An evaluation platform for general agents”. In: *Journal of Artificial Intelligence Research* (2013), pp. 253–279.

[14] Dimitri P Bertsekas et al. *Dynamic programming and optimal control*. 3. Athena scientific Belmont, MA, 2005.

[15] Gerald Tesauro. “Temporal difference learning and TD-Gammon”. In: *Communications of the ACM* 3 (1995), pp. 58–68.