Course Project: A Study of Curiosity-driven Exploration in Reinforcement Learning

Aleksei Petrenko
University of Southern California
petrenko@usc.edu

Abstract

One of the most apparent shortcomings of modern reinforcement learning methods is an underwhelming performance in environments with sparse rewards. A great portion of potential real-world applications for autonomous agents assumes the reward is given to the agent only upon accomplishing a task or reaching a goal. If it turns out to be impossible to stumble upon the rewarding state by mere random exploration, the situation for the majority of state-of-the-art RL methods is dire: without rewards, there is no learning signal and therefore no learning can occur. One possible solution to this problem is to modify the reward function to explicitly encourage the agent's curiosity about the environment. In the absence of external reward, this intrinsic motivation can help the agent to acquire skills that might be useful for solving the problem. In this project, we will explore a contemporary RL method that employs curiosity-driven exploration and evaluate its performance against the baselines on hard exploration tasks in a 3D pixel-based environment.

1 Introduction

Rapid progress in the field of reinforcement learning, especially in deep RL, has allowed reinforcement learning agents to achieve unprecedented results on a variety of control tasks, such as controlling the agent in the environments with very large state spaces (e.g. learning directly from pixels [1]) or continuous control for locomotion [2]. However, in many real-world scenarios, rewards extrinsic to the agent are very sparse or missing altogether. Under such circumstances, learning may become impossible, as the majority of contemporary methods rely solely on environment rewards for their policy updates. In practice, people will often try to counteract this effect with reward engineering: shaping the reward function by adding auxiliary goals or another form of dense reinforcement to help the learner. Very often reward engineering requires a lot of manual fine-tuning before the desired agent behavior can be achieved. It is therefore desirable to build algorithms that do not require careful tweaking in the sparse reward conditions and are able to discover rewarding states autonomously.

The vast majority of RL methods employ very simple exploration strategies, such as ϵ-greedy exploration, using high-entropy stochastic policies and sampling from the action distribution [3] or injecting Gaussian noise to actions. This, however, helps only in relatively simple environments. In reality the chances of visiting the rewarding state with random exploration can be astronomically low for a lot of problems.

Many of the classic, theoretically-proven methods that explicitly address exploration rely on counting visited state-action pairs. The idea is to turn this information into an auxiliary exploration reward, e.g. the well-known UCB algorithm [4] chooses actions that maximize the estimated objective plus the exploration bonus: $a_t = argmax_a \left(\hat{r_t} + \sqrt{\frac{2 \log t}{n(a_t)}} \right)$ where $\hat{r_t}$ is an estimated reward-to-go and $n\left(a_t\right)$ is the number of times this particular action has been chosen before. Some of the more advanced

methods take this idea further and use the counting of compressed representations or *hashes* of visited states [5].

Another line of work explores the idea of training the ensemble of policies. Even when trained on the exact same data, the models within the ensemble can learn diverse behaviors, which in turn can improve exploration. A successful example of this approach is the Bootstrapped DQN algorithm [6], that achieved great results on Atari games by training multiple Q-functions and switching between them when collecting experience.

One of the most promising approaches to exploration involves using *information gain* about the environment dynamics as an auxiliary exploration reward. For example, Variational Information Maximizing Exploration (VIME, [7]) utilizes Bayesian neural networks and defines exploration bonus as the KL-divergence between the predictive model weight distribution before and after the gradient update. VIME performs well on robotic locomotion problems with sparse rewards and thus demonstrates the potential of this idea.

In this project we will explore and evaluate another recently developed algorithm that uses the measure of uncertainty or *surprise* about the future as an exploration bonus. The formulation and details of the algorithm will be discussed in further sections.

2 Curiosity-driven Exploration by Self-supervised Prediction

This algorithm was originally proposed in 2017 by Pathak et al. [8]. The main idea behind this method is to use the error of the forward dynamics predictive model as an exploration reward bonus. This algorithm introduces Intrinsic Curiosity Module (ICM) as a reusable building block that can be applied to many different RL algorithms (actor-critic, policy gradient, DQN) and improve performance on hard exploration tasks. The main architecture of the algorithm is shown on the figure below:

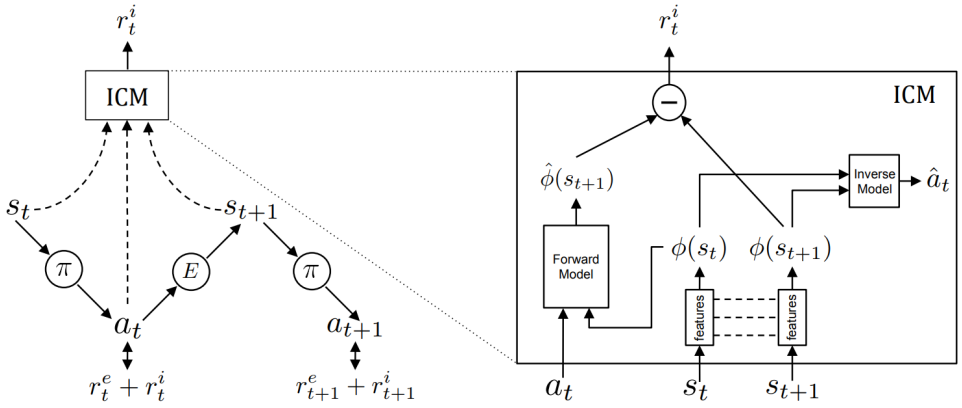


Figure 1: The agent in state s_t interacts with the environment by executing an action a_t sampled from its current policy π and ends up in the state s_{t+1}. The policy π is trained to optimize the sum of the extrinsic reward (r_t^e) provided by the environment E and the curiosity based intrinsic reward signal (r_t^i) generated by Intrinsic Curiosity Module (ICM). ICM encodes the states s_t, s_{t+1} into the features $\phi(s_t)$, $\phi(s_{t+1})$ that are trained to predict a_t (i.e. inverse dynamics model). The forward model takes as inputs $\phi(s_t)$ and a_t and predicts the feature representation of the next state $\hat{\phi}(s_{t+1})$. The prediction error in the feature space is used as the curiosity based intrinsic reward signal.

Note how ICM uses a pair of predictive models to generate the exploration bonus. One might think that it is possible to just use the forward dynamics model error as an exploration bonus, and indeed it might work for some environments. However, if the environment includes state transitions that are stochastic in nature or very hard to predict the agent will get stuck in a loop revisiting these transitions. A real-world example might be a waterfall: the exact dynamics of the water torrent are virtually impossible to predict, therefore the agent will get a high reward every time the waterfall is observed.

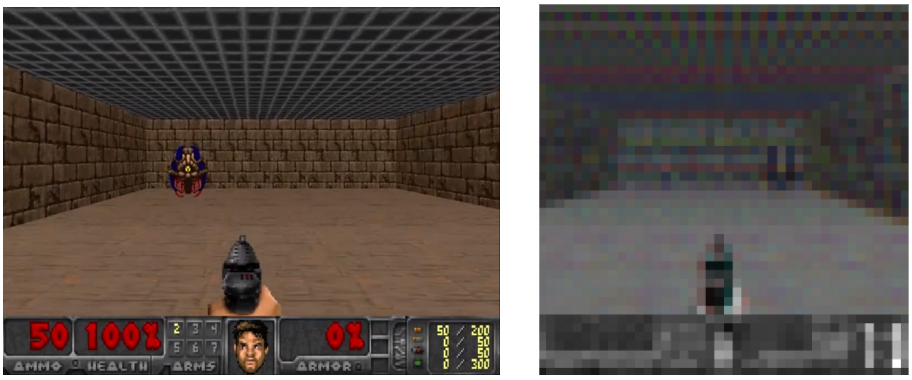
To counteract this, the algorithm uses features learned by the inverse dynamics model: a neural network that predicts an action conditioned on the pair of consecutive states. Then the exploration reward is calculated based on the prediction of this compressed representation $\phi(s_{t+1})$. Inverse dynamics model does not have an incentive to encode the information about the world that does not directly affect the agent or related to agent's actions. This architectural choice makes the algorithm robust to stochastic state transitions and it is able to continue useful exploration even if the exact value of s_{t+1} is very unpredictable in general.

3 Experiments

3.1 Environment

We evaluate the performance of the algorithm on a set of hard exploration problems in the 3D pixel-based environment ViZDoom [9], which is a clone of the well-known classic game Doom reimplemented for research purposes. This environment was wrapped in an OpenAI Gym-like [10] interface.

Environment details. In ViZDoom the agent is provided with the first-person view and is able to rotate and move freely in the 3D world. In the tasks that involve enemies the agent can also use the weapon. This results in a total set of 7 discrete actions that a policy must choose from. Standard tricks for pixel-based reinforcement learning were utilized to simplify the training process. The original ViZDoom environment outputs RGB frames in 160x120 resolution (Figure 2a), which are converted to grayscale and resized to 42x42 pixels (Figure 2b), thus requiring significantly smaller encoder convnet. The action generated by the policy is repeated in the environment 4 times, this helps to significantly reduce the number of forward passes through the policy network and therefore speeds up the training. Four consecutive environment frames are stacked as channels of the input image. This allows the feed-forward policy to capture the environment dynamics without the need for an RNN policy architecture.



(a) A typical frame from ViZDoom environment, resolution is 160x120x3. This particular environment where the agent is rewarded for shooting the enemy was used in a "doom simple" experiment.

(b) Resized and transformed input frame as seen by the neural network, resolution is 42x42x4 (four consecutive frames stacked together).

Figure 2: ViZDoom 3D learning environment.

3.2 Baseline RL algorithm

As a baseline algorithm we have chosen the synchronous variant of Advantage Actor-Critic algorithm. It is a policy gradient algorithm with a learned baseline [3, 11] which is similar to the algorithm used in the original paper [8]. In Advantage Actor-Critic (A2C) a set of workers interacts with the environment and collects the experience, which is then added to a minibatch and used for a

single policy update (which makes A2C an on-policy algorithm). The policy update is given by the following equation:

$$\theta \leftarrow \theta + \alpha\gamma^t \left(G_t - b(s_t)\right) \nabla \ln \pi \left(a_t|s_t, \theta\right) \tag{1}$$

which is a slightly modified REINFORCE update rule, where G_t represents the TD return, and $b(s_t)$ is a learned baseline parameterized by a neural network. In all experiments, the number of workers was 40 and the environment rollout was 5 steps, for a total batch size of 200. The policy entropy maximization objective was added to the total loss to make sure that baseline algorithm has some means for exploration. For the experiments with curiosity the A2C algorithm was augmented with an ICM module described in Section 2.

3.3 Exploration tasks

We evaluate the performance of the baseline algorithm and the curiosity-augmented algorithm on four problems defined in the ViZDoom environment. In the first experiment we use the most basic entry in the ViZDoom problemset, where the agent must shoot the enemy standing on the other side of the room (Figure 2a). This is a relatively simple problem with dense rewards and by comparing vanilla A2C with curiosity-augmented algorithm we observe how the bonus exploration objective affects the learning. We refer to this task as "doom simple".

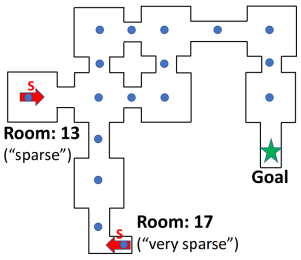
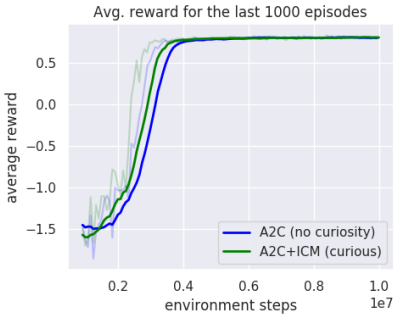


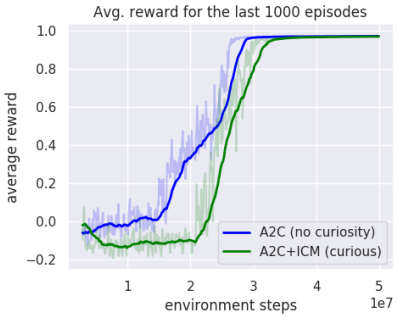
Figure 3: Testing maze. Green star - goal location. Blue dots refer to agent spawning locations in the "maze basic". Rooms 13, 17 are the fixed start locations in "sparse" and "very sparse" reward cases.

We conduct our three main experiments in ViZDoom maze environments of increasing difficulty (Figure 3). We call these environments "maze basic", "maze sparse" and "maze very sparse". In all mazes the agent receives +1 reward for collecting a goal and small negative reward every step otherwise (thus encouraging the agent to reach the goal quickly). The episode is terminated after 2100 steps if the goal is not reached. In "maze basic" the spawn location of the agent is sampled randomly in every episode, and the agent is therefore able to occasionally reach the goal following just random exploration. In "sparse" and "very sparse" settings the spawn locations are fixed (far away from the goal), which makes the problem much harder. Both "sparse" and "very sparse" environments require hundreds of deliberate actions to reach the goal from starting position, which makes its random discovery very unlikely.

Figures 4a and 4b demonstrate that addition of curiosity module does not significantly affect performance on tasks with dense rewards, although minor decrease in performance can be expected as the agent is "distracted" by auxiliary rewards. Figures 5 and 6 show the clear advantage of curiosity-based exploration in environments with very sparse rewards.



(a) Results on "doom simple".



(b) Results on "maze basic".

Figure 4: Results on dense reward tasks: addition of curiosity module does not significantly affect performance.

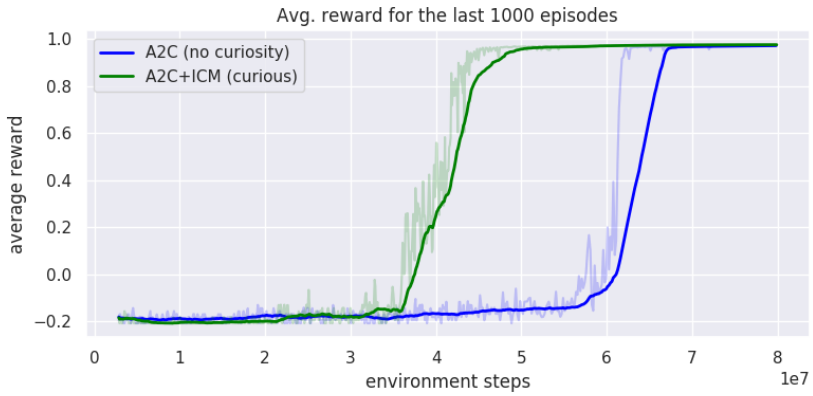


Figure 5: Results on "maze sparse": with curiosity the agent is able to learn much faster.

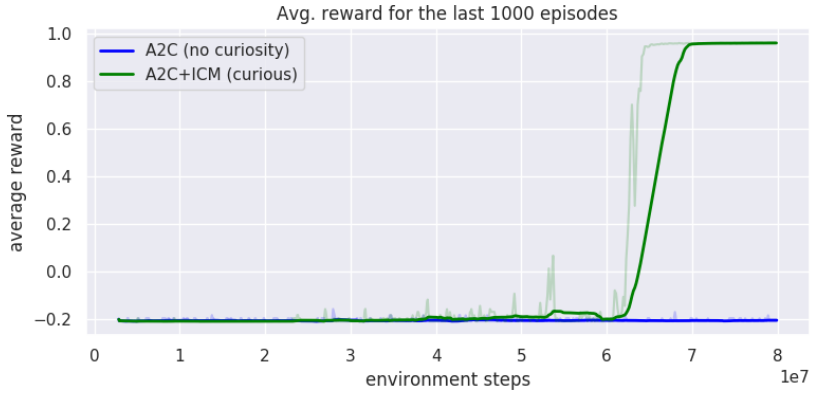


Figure 6: Results on "maze very sparse": only the curious agent is able to eventually achieve the goal.

4 Conclusion

The experiment results indicate that using a measure of "surprise" about the future, as it is done in Curiosity-driven Exploration by Self-Supervised Prediction, is a good approach to hard goal-directed exploration problems. The method significantly outperforms the baseline algorithm and is able to solve environments where standard methods fail. As a downside, this method introduces large number of hyperparameters, such as learning rates of inverse and forward dynamics models, scaling factor of the reward bonus, architecture details of predictive models, etc. The algorithm has proven to be hard to tune because all of the parameters have to be chosen just right. Indeed, recent articles on curiosity-based exploration suggest that this method can be simplified and improved significantly [12, 13].

The curiosity-based learning is one of the strongest known approaches to exploration-exploitation dilemma in deep RL and has a great potential for further research. Without a doubt one day we will see similar ideas implemented in the next generation of highly advanced AI systems.

Resources. The algorithm was implemented in Python & Tensorflow, the source code can be found at: https://github.com/alex-petrenko/curious-rl. Video demonstration of the agent performance is available on YouTube: https://youtu.be/ufhih7jTuX4.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[3] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[4] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, March 1985.

[5] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. *CoRR*, abs/1611.04717, 2016.

[6] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *CoRR*, abs/1602.04621, 2016.

[7] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *CoRR*, abs/1605.09674, 2016.

[8] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017.

[9] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. The best paper award.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[11] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[12] Yuri Burda, Harrison A Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *CoRR*, abs/1810.12894, 2018.

[13] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *arXiv:1808.04355*, 2018.