Deep Curiosity Networks

Chris Chute
Stanford University
chute@stanford.edu

Erik Jones
Stanford University
erjones@stanford.edu

Abstract

We introduce Deep Curiosity Networks (DCN), a technique for guiding exploration of reinforcement learning agents towards states that both novel and useful for predicting future observations. In DCN, an agent receives an exploration bonus that is proportional to the product of a scalar *curiosity score* and the error of a neural network trained to predict features of the next observation. The curiosity score approximates how useful an observation will be for training the predictor network: *i.e.,* the curiosity score is the estimated increase in accuracy of the next-observation predictor network after being trained on an observation. To make DCN tractable for large-scale training, we parameterize the curiosity score with a neural network trained offline. We analyze the performance of agents trained with DCN on Montezuma's Revenge, an Atari game that is a benchmark for exploration methods due to its sparse rewards.

1 Introduction

In reinforcement learning, an agent learns by taking actions from different states and observing rewards. As the state space becomes more complex, however, it becomes intractable to visit and thus learn about every every possible state. One method that works well in large state spaces is deep Q-Learning, where the Q-value function is parameterized by a deep neural network [1]. Deep Q-Learning has led to improvement of the state of the art in many RL problems [2, 3, 4], and makes developing large scale generalizable RL systems more feasible.

Online algorithms for RL must trade off between *exploration* and *exploitation*. For all but the most trivial tasks, long-term rewards come at the cost of short-term sacrifices. Without exploration, an agent will never discover these long-term rewards and will instead settle into a suboptimal policy. In problems with high-dimensional states and sparse rewards, exploration is especially difficult due to the large state space and long gap in time between exploration and a resultant reward. Montezuma's Revenge is one such environment with a high-dimensional state space (*e.g.,* we use $2^{8 \cdot 84 \cdot 84}$ grayscale pixel configurations) and sparse rewards (*e.g.,* collecting rare keys). As a result, Montezuma's Revenge has become a benchmark for exploration algorithms. The relative sparsity of extrinsic rewards makes it advantageous to define an *intrinsic reward*, or a reward that incentives visiting novel and informative states, even if they don't provide a reward in the original game.

In this work, we propose a new exploration bonus called Deep Curiosity Networks (DCN). In DCN, an agent is trained to maximize the usual extrinsic reward in addition to an intrinsic reward based on the *curiosity score* of an observation. The curiosity score is parameterized by a deep neural network (the *curiosity network*), which is trained offline to predict the improvement of a separate predictor network in predicting random features of the observation. We integrate DCN with Random Network Distillation [5], a recently proposed state-of-the-art exploration strategy, and we evaluate the algorithm's performance on Montezuma's Revenge.

Preprint. Work in progress.

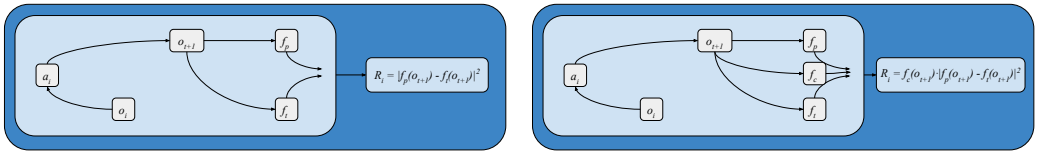

Figure 1: *Left:* Vanilla RND, in which the intrinsic reward is the error of a neural network f_p in predicting the outputs of a fixed random network f_t. *Right:* Our proposed method, RND with DCN, where f_c is the curiosity network, trained offline to predict improvement in f_p. Figure inspired by [6].

2 Related Work

Exploration methods for reinforcement learning can be roughly categorized into undirected and directed exploration methods [7]. In deep reinforcement learning with high-dimensional inputs, many directed methods such as counter-based exploration [8], competence maps [9], and dynamic programming [10, 11] are intractable. As a result, the pioneering works in deep RL for Atari used a simple undirected ϵ-greedy strategy [12, 13].

Undirected methods are inferior to directed methods in terms of efficiency of exploration [14]. Although simple undirected approaches often suffice for environments with dense rewards, more powerful directed approaches are necessary for environments where rewards are sparse. In the Atari domain, Bellemare et al. identified the games in which sparse rewards make exploration a challenging task, one such game being Montezuma's Revenge [8]. For example, the original DQN failed to achieve any reward on Montezuma's Revenge, and later works explicitly focusing on directed exploration methods have also received zero reward in the game [15].

Recently, many advances have been made in exploration strategies on Montezuma's Revenge. For example, RL agents have made significant progress when given access to the emulator state [16, 17] or expert demonstrations [18, 19, 20]. Random Network Distillation (RND) [5] is a recent technique that requires neither emulator state nor expert demonstrations, yet achieves state of the art on Montezuma's Revenge.

RND uses a fixed random neural network to encode features of the next observation, and trains a second predictor network to match those features. The error of the predictor network serves as an intrinsic reward for the agent, thereby encouraging exploration. More formally, if we define f_p to be the predictor network and f_t to be the randomly initialized target network, then the intrinsic reward R_{int} is defined as

$$R_{\text{int}}(o_{t+1}) = \|f_p(o_{t+1}) - f_t(o_{t+1})\|_2^2 \,.$$

The target network stays fixed at its random initialization, but the predictor network is trained to minimize $\mathcal{L}_p = R_{\text{int}}$, *i.e.,* to match the target network's outputs. Less familiar observations tend to give larger error and thus higher intrinsic reward, leading to exploration of novel states. A schematic of RND is provided in Figure 1.

It is worth noting that similar methods prior to RND (*e.g.,* [15]) used predictor networks to map $(o_t, a_t) \to o_{t+1}$, rather than RND's formulation of $o_{t+1} \to f_t(o_{t+1})$. The main weakness of the former approach is the *noisy TV problem*, in which agents fixate permanently on states with stochastic transitions. RND's formulation addresses the noisy TV problem by training the predictor to match the deterministic feature map f_t.

Although RND does not suffer from the noisy TV problem, the proposed fix distances the algorithm further from biological inspirations for curiosity. Developmental analyses of curiosity-driven exploration have pointed out that intrinsic reward should be a combination of novelty *and* usefulness in improving future predictions [21]. Therefore, methods such as RND which use prediction error are theoretically suboptimal in that they reward novelty but not usefulness of novel observations. Our work is an attempt to address this concern in the context of RND.

3 Dataset

We collect a training set of roughly 2 million examples for offline training of the curiosity network. Each example consists of a pair (o_t, i_t), where o_t is the observation at time t and i_t is the improvement of the predictor network f after being trained on a mini-batch of observations including o_t. That is,

$$i_t = \|f_p(o_{t+1}) - f_t(o_{t+1})\| - \left\|f_p^*(o_{t+1}) - f_t(o_{t+1})\right\|,$$

where f_p^* is the predictor network after updating f_p with a small inner-loop of gradient descent steps. Since f_p^* is already computed during the normal execution of RND, collecting this dataset adds negligible overhead to the algorithm's runtime.

The observations o_t are frames preprocessed as in [12]: Four sequential frames are sampled from the emulator, downsized to 84×84, and combined by taking the pixel-wise maximum across the frames. Four such samples are stacked together and converted to grayscale to create each o_t. To stabilize training of the curiosity network, we convert the i_t values to three classes, corresponding to the improvement ranges $(-\infty, 7 \cdot 10^{-5})$, $[7 \cdot 10^{-5}, 5 \cdot 10^{-4})$, and $[5 \cdot 10^{-4}, \infty)$. The boundaries were determined to partition the dataset roughly into thirds, and intuitively the three classes correspond to "not curious," "neutral," and "curious" examples. Thus the final (o_t, i_t) pairs were pre-processed frames of shape $84 \times 84 \times 4$ and one-hot class labels of size 3.

4 Methods

There are two different components of our system: the training of the curiosity network and the application. We discuss both in further detail.

4.1 Curiosity Network Training

Before adding a curiosity score to Montezuma's Revenge, we first train a *curiosity network* offline. The goal of the curiosity network is to predict the change in the error of the original predictor network given the current observation. As described in the dataset section, observations mapped to the change in error were stored by running through Montezuma's Revenge using the original RND. Given this data, two variables remain: the architecture of the curiosity network and the output.

4.1.1 Architectures

We used linear regression as a baseline for the curiosity network. Because the inputs to the network are preprocessed screenshots, they are three dimensional arrays. To execute linear regression, we simply flatten the image into a single dimensional vector and pass it through a layer mapping the input dimension to the number of classes.

Linear regression, however, loses a significant amount of the spatial information of the frame. To take advantage of the fact that our input is an image, we additionally try a residual neural network for our curiosity network [22]. Residual neural networks add shortcut connections to a vanilla convolutional neural network, which reduces the vanishing gradient problem. ResNets have been shown to perform well on ImageNet among other imaging tasks, suggesting it could function well as a curiosity network.

4.1.2 Predictions

The natural output from the curiosity network would match the saved curiosity score, which we calculated by finding the change in error of the original predictor network after observing some observation. This frames our problem as a regression task, which we attempt as a baseline.

However, in practice essentially all of these curiosity scores are close to zero. By framing the problem as a regression problem with mean squared error, the model can more or less continuously approach the zero function while the error monotonically decreases. Small differences in curiosity score, however, can correspond to significant differences in optimization. We solve the problems with regression by instead framing the curiosity score as a multi-class prediction problem with discretized boxes. After examining the distribution of curiosity scores, we map scores less than $7 \cdot 10^{-5}$ to bucket 0, scores greater than $5 \cdot 10^{-4}$ to bucket 2, and everything in between to bucket 1. In this formulation, our curiosity model outputs three predictions, which are then put through a softmax layer to produce probabilities for each bucket.

4.2 Curiosity Network Implementation

After training the curiosity network, we must deploy it within the current RND model. To do so, we take the intrinsic reward from the RND formulation, which is the error of the RND predictor network, and scale it by some function of the curiosity score. In the case where we run regression, we get the distribution of a sample of the model outputs and then map that distribution to a Gaussian with mean 1 and variance 1. By mapping to a mean one distribution, we ensure that the expected intrinsic reward in our modified network is the same as the original in RND, but observations with higher curiosity scores lead to greater intrinsic rewards.

In the discrete case, we need a function that maps a triple of probabilities (p_0, p_1, p_2) where p_i corresponds to the probability the curiosity outputted by the observation is within the defined range of bucket i. If $p = (p_0, p_1, p_2)$ and $\theta = (0, 1, 2)$, We define our function $f : \mathbb{R}^3 \mapsto \mathbb{R}$ as follows:

$$f(p) = \theta^T p$$

It's easily shown that the expectation of $f(p)$ if p is a valid probability (i.e. sums to one and is non negative) is one. Moreover, using our choice of f, the difference between $f(p)$ and 1 (and thus the amount of required adjustment to the curiosity network) is $p_2 - p_0$, which implies that normal RND is altered by a factor of of the difference in probabilities of the extremes.

5 Results

We present results on training an agent with RND and DCN on Montezuma's Revenge. The intrinsic reward is closely related to the agent's exploration as measured by total number of rooms visited. This phenomenon is shown in Figure 2: When the agent visits a new room, the intrinsic reward spikes sharply. Just after entering a new room, the intrinsic reward is consistently higher than when the agent has thoroughly explored the known set of rooms.

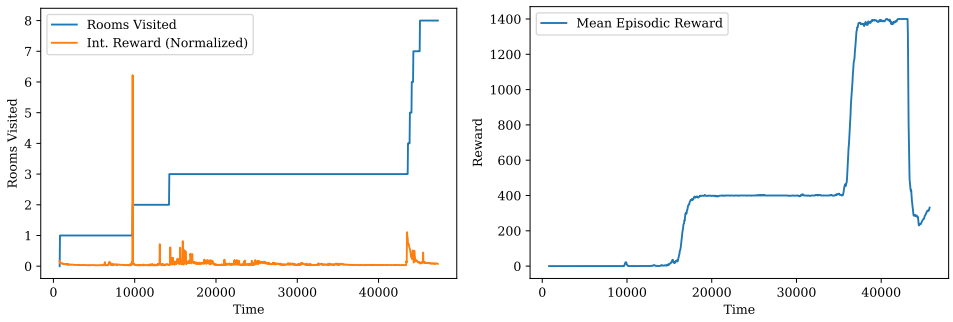

Figure 2: *Left:* Intrinsic rewards with DCN are closely related to exploration of new rooms. *Right:* Mean episodic reward vs. time with DCN.

Due to computational constraints, our experiments were not as long-running as those in RND, so a fair, full comparison with pure RND is not possible at this time. Most of the numbers reported in RND are running 1024 parallel environments across 8 GPUs for 500k timesteps. By comparison, we were only able to run 384 environments across 3 GPUs for 50k timesteps. Even with less than half the number of parallel environments, we were able to visit 8 rooms inside of the first 50k timesteps, seeming to match RND's reported numbers (*e.g.,* Figure 5 in [5]).

6 Discussion

6.1 Curiosity Evaluation

Before we start our more comprehensive analysis, it's worth considering whether or not our formulation for curiosity: comparing the error between subsequent steps of the original predictor network, is actually capturing something reasonable within the mechanics of Montezuma's Revenge. In Figure 1, we extract ten observations from to the top hundred curiosity scores out of more than 3000 samples. We see that in each of these ten frames, the player is learning something useful about the game,

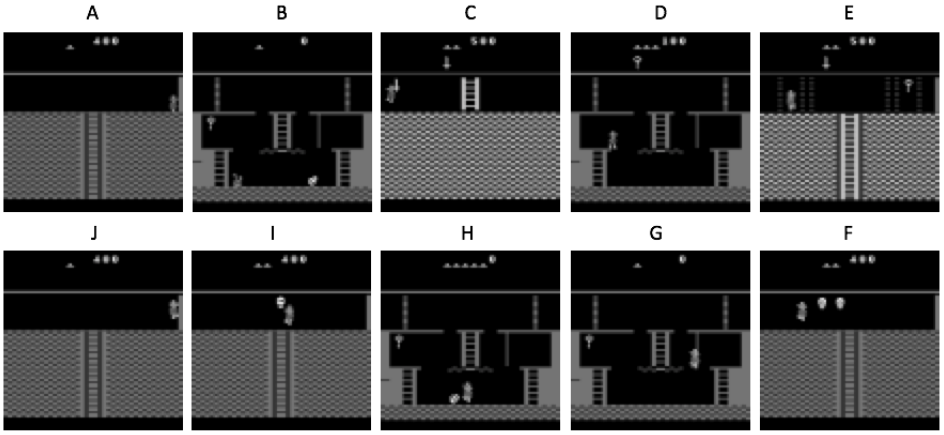

Figure 3: Observations from Montezuma's Revenge with the highest curiosity scores (10 samples from the top 100 observations, out of over 3000 examples). In A. the player is about to enter a new room. In B. the player fell and is about to die. In C. the player is jumping to obtain a key. In D, the player is about to fall of the ledge to his death. In E. the player must time a forward run correctly to get past electric shocks. In F. the player must learn the dynamics of the two skeletons to avoid them and stay alive. In G. the player grabs a rope and learns it doesn't fall. In H. the player is about to die from a bouncing skeleton on the ground. In I. the player is about to hit a skeleton in the air, and in J the player is once again about to change rooms.

ranging from near death experiences, to rewards, to simple game mechanics. In each of these cases we want a high intrinsic reward, which our curiosity score seems to be capturing.

6.2 Training Time Comparison

One main selling point of DCNs is that by providing better intrinsic rewards it can reduce overall training time. Training time, however, can refer to three things: either the total time it takes for a DCN to get to some arbitrary point in Montezuma's Revenge, the total time an agent spends exploring Montezuma's Revenge (including both the RND data curation time and the DCN runtime), or the total time involved in training a DCN including the offline curiosity network training time. Depending on the choice, curiosity networks can offer varying performance relative to RNDs. The choice of metric should vary based on the application: in cases where access to the environment is limited but other resources are not, the offline component of the DCN training process could add utility without more environment queries. Moreover, RNDs tend to converge to some optimal performance. If computation power is not a constraint, the varying components of the curiosity networks could lead to better asymptotic convergence.

7 Future Work

7.1 Experiments

As mentioned in the previous section, due to limits on the types of GPUs available and the time with them, There are a number of experiments that would be natural extensions to run. The most basic ones involve

- Running DCNs and RND on a rig that is capable of handling both unmodified for a set amount of time, and comparing performance
- Comparing asymptotic performance between DCNs and RND
- Comparing different time ratios for DCNs (i.e. time to curate data, time to train the curiosity network, and time to learn a policy for Montezuma's Revenge
- Trying DCNs on games other than Montezuma's Revenge

With more continuous computation time, a more rigorous evaluation of DCNs becomes more feasible.

7.2 Curiosity Network Insertion

As implemented, we insert the Curiosity Network into the RND training procedure by scaling the intrinsic reward obtained from RND. In essence, we preserve as much about RND is possible: by scaling we don't introduce any new terms, and our mapping from the output probabilities to the scaling constant has expected value 1, again mimicking the RND approach.

There's little indication, however, that the constant used to scale the RND predictor network in vanilla RND should be one. In the case where rewards are only intrinsic, the scaling factor is of course arbitrary, but a good experiment to run in practice would be to optimize a function mapping the expected extrinsic reward from the game to the intrinsic reward constant. This would help make RND more generalizable, and similarly improve performance of the corresponding DCNs.

Another potential approach that we did not explore is adding a separate intrinsic reward corresponding to the curiosity score, as opposed to scaling. The effect of adding a separate intrinsic reward serves to increasingly reward lopsided observations, where one of the curiosity score and the RND intrinsic reward is high where the other is low. This is worth testing, but makes the model more susceptible to noise in the curiosity score and the RND intrinsic reward, which would be ironic given DCNs were originally designed to combat the RND "noisy tv" problem.

7.3 Different Curiosity Methods

While we have shown that the change in error of the original predictor network captures many of the characteristics we would hope for in a curiosity score, it's conceivable there's something better. One more direct optimization to RND would involve defining the curiosity score by the change in error of the RND predictor network. While this is less likely to improve RND in places where the intrinsic rewards are bad, it could help RND train faster, especially for environments where heavy computational resources are required at each iteration.

7.4 Online Curiosity Network Training

If runtime is a significant concern and different periods of the game have similar characteristics (i.e. in Montezuma's Revenge, you always want to avoid skeletons) training the curiosity network online could lead to runtime improvements later in the training process. In an online approach with predicted curiosity score c and RND bonus r, one could define the intrinsic reward IR as:

$$IR = \epsilon c + (1 - \epsilon)r$$

Where in this formulation ϵ starts at 1 and decreases by some prescribed method. This approach would involve some funky Tensorflow mechanics, however, as the curiosity network would need to continually update.

7.5 Probability Mapping Function

Lastly, while our function mapping the probabilities for three classes into a curiosity score had some desirable properties, it's far from the only function possible. Particularly if the original intrinsic rewards for RND are too small or too large, experimenting with different expected values for this probability function could improve performance. Though this would add training complexity, one could even learn a mapping from probabilities to the curiosity score separately, limiting the amount of human involvement.

8 Contributions

E.J. and C.C. combed through the original OpenAI implementation of Random Network Distillation together. E.J. then wrote the original training script for the curiosity network, and then added checkpoint saving, checkpoint recovery, and TensorBoard integration. C.C. wrote the ResNet implementation of the curiosity network, and developed the pipeline to extract the data necessary to train the DCN. C.C. and E.J. worked together to integrate the curiosity network into RND training, which became an extensive process.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, 02 2015.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016.

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

[4] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, pages 50–56, New York, NY, USA, 2016. ACM.

[5] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[6] Reinforcement learning with prediction-based rewards. `https://blog.openai.com/reinforcement-learning-with-prediction-based-rewards/`. (Accessed on 12/07/2018).

[7] Roger McFarlane. A survey of exploration strategies in reinforcement learning. *McGill University, http://www. cs. mcgill. ca/ cs526/roger. pdf, accessed: April*, 2018.

[8] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[9] Sebastian B Thrun and Knut Möller. Active exploration in dynamic environments. In *Advances in neural information processing systems*, pages 531–538, 1992.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[13] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[14] Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

[15] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[16] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2753–2762, 2017.

[17] Christopher Stanton and Jeff Clune. Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. *arXiv preprint arXiv:1806.00553*, 2018.

[18] Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado van Hasselt, John Quan, Mel Večerík, et al. Observe and look further: Achieving consistent performance on atari. *arXiv preprint arXiv:1805.11593*, 2018.

[19] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

[20] Michał Garmulewicz, Henryk Michalewski, and Piotr Miłoś. Expert-augmented actor-critic for vizdoom and montezumas revenge. *arXiv preprint arXiv:1809.03447*, 2018.

[21] Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.