Unsupervised pre-training in Reinforcement Learning
6.883 Project Report

Rohil Verma Paolo Gentili Martin Schrimpf

Abstract

Most techniques in Reinforcement Learning assume frequent and inexpensive rewards. In real-world settings such as robotics, rewards are however expensive to obtain. Here, we investigate the benefit of pre-training agents in an environment free of rewards, for the purpose of obtaining skills that later speed up training in the reward-setting. In particular, we compare pre-training of the agent's perception module and curiosity-driven pre-training of the whole agent. We find that, using the same hyper-parameters that work for extrinsic training alone, neither approach reliably speeds up training. However, curiosity-driven pre-training allows the agent to achieve maximum performance on *VizDoom* whereas no pre-training fails to train. Our findings suggest that while pre-training may improve training in some cases, it does not reliably yield a solution.

1 Introduction

Agents in Reinforcement Learning typically operate in environments where rewards are readily available, such as continuous running scores in Atari games (Mnih et al., 2013a), or distances between a robot arm and a target object (Lillicrap et al., 2015). In the real world however, rewards might be very sparse (Pathak et al., 2017), not available at all times or difficult to obtain, due to the costs associated with rewarding the agent when human annotators are involved (Pathak et al., 2018). In robotics for instance, humans often have to provide carefully crafted examples of successful movement trajectories (Gu et al., 2016).

Outside of Reinforcement Learning, there have been numerous approaches to learn models of the world without external supervision: auto-encoders (Olshausen and Field, 1997; Diederik P Kingma and Welling, 2014) learn a compact feature representation by encoding the input into a latent representation and then decoding the original input again. Generative Adversarial Networks (Goodfellow et al., 2014) focus more on generative tasks but also learn a latent representation of the input distribution.

All of our code is available at `https://github.com/mschrimpf/unsupervised-rl-pretraining`.

In Reinforcement Learning, extrinsic rewards are often enriched with an intrinsic exploration strategy that maximizes the information gain of the agent's current understanding of its environment (Houthooft et al., 2016). Klyubin et al., 2005 and Mohamed and Jimenez Rezende, 2015 consider the information gain based on a potential sequences of actions. Other works examine exploration using count-based methods, in terms of the number of states that have been visited (Bellemare et al., 2016; Tang et al., 2016). However, there are only very few results from training in an entirely rewards-free environment (Pathak et al., 2017; Haber et al., 2018).

In this work, we investigate the case where rewards are difficult to obtain but the agent can prepare itself by pre-training in the same environment or elsewhere. Our goal is to outline techniques that might be useful to pre-train an agent and later reduce its training time in the environment with rewards where training might be costly. Namely, we compare (1) an agent trained in the target environment with rewards right away, (2) training the agent's perception module on a classification task beforehand, and (3) training the agent in the target environment with only intrinsic but no extrinsic rewards. For each setting, we quantify the amount of time it takes the agent in the extrinsic reward setting to achieve maximum performance. We find that in many of the environments investigated here, pre-training does *not* benefit later fine-tuning with full extrinsic rewards even though the same hyper-parameters let the agent train successfully without pre-training. In the VizDoom environment however, the agent trained *only with pre-training*: agents without pre-training failed to reliably perform altogether; perceptual pre-training got performance to low levels; and curiosity-driven pre-training let the agent train to maximum performance.

2 Reinforcement Learning

In Reinforcement Learning, an agent in a current state s_t takes an action a_t from a set of possible actions A. The agent then receives a scalar reward r_t, but this reward might be sparse and zero for most of the actions. The agent therefore has to relate its rewards to actions and determine which of its actions led to the reward. Given a state s_t, the agent therefore chooses action a_t based on a learned policy π. This process runs until the environment signals a final state (e.g. game over, level complete) after which one episode is over and everything starts anew. At each time step t, the total accumulated return is denoted by $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ where γ is a discount factor $\in (0, 1]$. The goal of the policy is to maximize the expected return from each state s_t.

Mapping from state s_t to action a_t directly is difficult without underlying knowledge of the expected return. Therefore, an action value $Q^\pi(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t]$ yields the expected return for selecting action a_t in state s_t under a given policy π. The value function $Q^*(s_t, a_t) = max_\pi Q^\pi(s_t, a_t)$ represents the maximum action value achievable by any policy for state s_t and action a_t. Policy π is then simple: to maximize the total expected return, the best choice is the action a_t^* that maximizes the current expected return under $Q*^\pi(s_t, a_t^*)$.

In deep Reinforcement Learning, the policy π and/or action value $Q^\pi(s_t, a_t)$ are generally represented using a neural network with parameters θ. θ is optimized to maximize the expected sum of rewards $\mathbb{E}_{\pi(s_t;\theta)[\sum_t r_t]}$ (Pathak et al., 2017; Mnih et al., 2016).

A3C Mnih et al., 2016 discovered that so-called "asynchronous advantage actor critic policy gradient" (*A3C*), parallelized agent-learners, stabilized training and allowed Atari training on multi-core CPUs instead of a GPU. In brief, multiple agents train in parallel (asynchronously), but parameters are shared following an update rule, removing communication costs of sending gradients and parameters. Multiple agents training in parallel stabilize the training since they are likely exploring different parts of the policy-space. Parallelization further reduces training time, roughly linearly with the number of parallel agents. Experience replay (Riedmiller, 2005; Mnih et al., 2013b; Van Hasselt et al., 2016; Schulman et al., 2015a; Schaul et al., 2016) is no longer required since learning is stabilized through diverse exploration. This allows on-policy reinforcement learning methods to be used which are simpler to implement and train (Mnih et al., 2016).

Specific architecture Our agent starts by extracting features with a perception module: four convolutional layers with 32 filters each, kernel size of 3×3, stride 2 and padding 1. Each convolutional layer is followed by an exponential linear unit (ELU, (Clevert et al., 2015)). The output of the perception module, i.e. the last convolutional layer, is the input to an LSTM with 256 hidden units. The value function Q and action a_t are then predicted with two separate fully-connected layers from the LSTM features.

3 Unsupervised pre-training

In this section, we detail two approaches to pre-train a Reinforcement Learning agent in an unsupervised manner, i.e. without external labels from the environment. The first version, *Perceptual pre-training*, trains only the agent's perception module outside of the environment whereas the second version, *Curiosity-driven pre-training* trains the whole agent inside the environment, but without extrinsic rewards.

3.1 Perceptual pre-training

Perceptual pre-training trains only a part of the agent, namely its perception module. This is done outside of the environment in order to quickly obtain a usable feature representation so that the agent has to spend less time in an environment with rewards. The learned feature transformations could then hopefully be re-used for the full agent in its environment and give it a head-start by providing useful features right away that would otherwise have to be learned in the environment with extrinsic rewards.

Specifically, we take the first four convolutional layers of the agent with weights θ_p, extend them with a fully-connected layer with 10 outputs and weights θ_f, and train the whole network with parameters θ_p and θ_f on an image classification dataset. Here, we use CIFAR-10 (Krizhevsky, 2009), a dataset of 60,000 32×32 color images divided into 10 classes.

After training, we throw away the fully-connected layer making the class predictions and re-use only the convolutional layers with weights θ_p in the agent. The weights for other layers are initialized randomly, as if the agent were trained from scratch. However, they hopefully adapt quickly to make effective use of the pre-trained feature transformation.

3.2 Curiosity-driven pre-training

While perceptual pre-training trains only part of the parameters, curiosity-driven pre-training (Pathak et al., 2017; Haber et al., 2018; Schmidhuber, 2010) trains all of the network's weights θ. The agent may be trained in a different environment or in the environment in which it will later be fine-trained right away. However, the agent does not yet receive extrinsic rewards (such as game scores) from the environment. This is meant to represent a case in which the agent can explore its environment, but rewards or labels are expensive and therefore not yet available. In order to minimize the potentially expensive process of obtaining labels, the agent explores the environment as much as possible to learn perception module and motor skills beforehand. When extrinsic rewards then become available, the agent can already access a rich feature representation and a repertoire of skills.

In particular, the agent jointly learns a forward model for the feature space $\phi(s_t)$ predicting $\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F)$ as well as an inverse model predicting the action $\hat{a}_t = g(s_t, s_{t+1}; \theta_I)$ restricting information gain to only parts of the environment that the agent can influence. The agent then chooses an action a_t to maximally fool its world model such that uncertainty is explored and further minimized. This is done by expressing the loss as a combination of the forward model and the inverse model. The loss for the forward model minimizes the difference in predicted next state $\phi(s_{t+1})$ and actual next state $\hat{\phi}(s_{t+1})$:

$$L_F(\phi(s_t), \hat{\phi}(s_{t+1})) = \frac{1}{2}||\hat{\phi}(s_{t+1}) - \phi(s_{t+1})||_2^2$$

For the inverse model, the loss L_I measures the discrepancy between the predicted and actual actions, namely maximum likelihood estimation under a multinomial distribution in the discrete case. Combining the forward and inverse loss, the agent overall optimizes

$$min_\theta\Big((1-\beta)L_I + \beta L_F\Big)$$

where $\beta \in [0, 1]$ weighs the inverse against the forward loss (Pathak et al., 2017).

4 Game Environments

In order to investigate the effects of these different pretrainings on the eventual performances of the agent, we run experiments in several game environments available through OpenAI's Gym (Brockman et al., 2016). In particular, we consider the *VizDoom* environment, along with several *Atari* games.

4.1 VizDoom environment

VizDoom (Kempka et al., 2016) is a platform based off of the first-person shooter game Doom, modified for use in visual reinforcement learning. As in the paper Pathak et al., 2017, we experiment with the 'DoomMyWayHome-v0' environment, which involves navigating in 3D to find a vest hidden in one of many interconnected rooms. The agent receives an RGB $480 \times 640 \times 3$ pixel input, and can choose to either move forward, turn right, turn left, or take no action as it navigates the course. The agent receives a reward of $+1$ if it finds the vest within the allowed time frame, 2,100 time steps, and 0 reward otherwise.

4.2 Atari environments

We additionally test these learning techniques with several Atari games, so as to view their effects on visually simpler games commonly used in reinforcement learning. In particular, we examine the games *Pong*, *Breakout*, and *Montezuma's Revenge*. The agent receives a size $210 \times 160 \times 3$ pixel input, and chooses from a small finite set of actions, specific to the game. As with the traditional DeepMind Atari paper (Mnih et al., 2013a), we use the deterministic game environments, in which each chosen action is repeated for exactly four frames. The agent's reward is also game-specific, corresponding to the increases in score that a human player would receive.

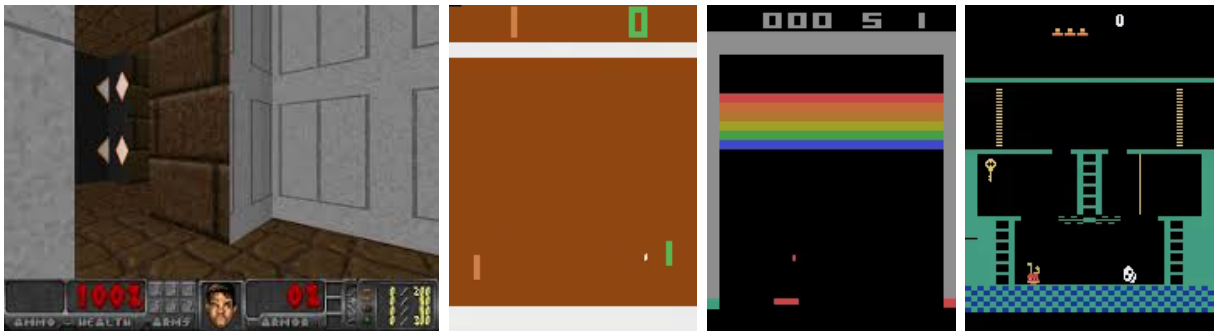

Figure 1: **We evaluate our agents in *VizDoom, Pong, Breakout* and *Montezuma's Revenge.*** The Atari environments under consideration span simple single-screen 2D games where everything is observable (Pong, Breakout) as well as more complex 2D and 3D games where the agent moves through the environment and thereby modifies its current field of view (Montezuma's Revenge, VizDoom).

5 Results

In the following, we evaluate *VizDoom, Pong, Breakout*, and *Montezuma's Revenge* without any pre-training, with perceptual pre-training and with curiosity-driven pre-training.

5.1 No pre-training

We train the agent in each of our environments using the same training scheme as in Pathak et al., 2017, using the Adam optimizer (Diederik P. Kingma and Ba, 2014) to update weights based on purely extrinsic rewards.

For all environments, we use a discount factor of $\gamma = 0.99$ and $\lambda = 1$ (for Generalized Advantage Estimation, see Schulman et al., 2015b), with an initial learning rate of 10^{-4}. We use policy rollouts of length 20.

The results are summarized in Figure 2. Training *VizDoom* on purely extrinsic rewards led to no noteworthy increases in performance, across slightly more than 10 million training steps. For *Pong*, the agent fails to improve its performance for approximately 2 million global training steps, at which point it quickly improves, plateauing at near optimal performance (of reward 20) after roughly 4.5 million global steps. In *Breakout*, the agent steadily improves its

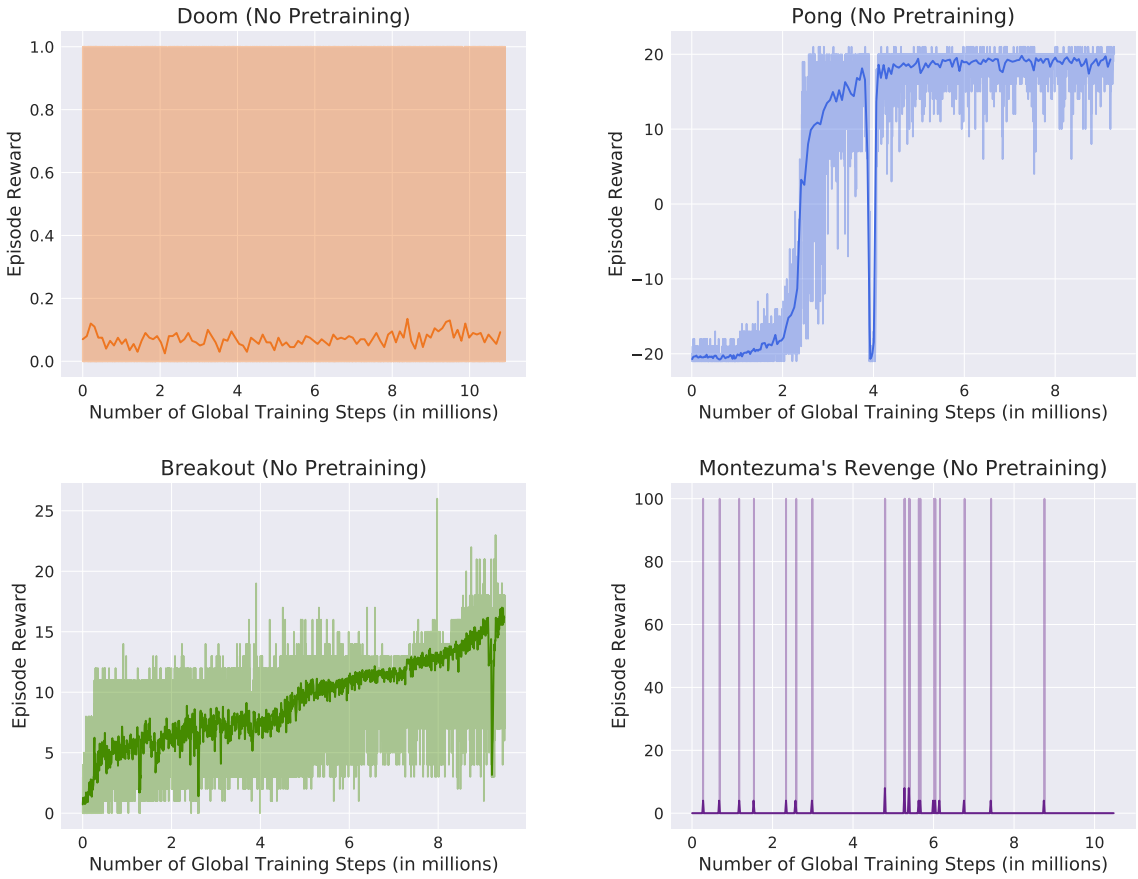

Figure 2: **Networks without pre-training, trained directly with A3C.** When training with extrinsic rewards directly, we observe that the number of epochs required for training varies from environment to environment: while some models require over 10 million steps to train, some learn more quickly, and some fail to learn at all.

performance throughout the training, achieving its highest rewards of approximately 17 after roughly 10 million steps. In *Montezuma's Revenge*, the agent fails to exhibit any notable increase in performance throughout the 10 million training steps.

5.2 Perceptual pre-training

The perception module was pre-trained to around 80% training performance on CIFAR-10. We trained the convolutional layers of the agent together with a dense classification layer on CIFAR-10. Networks were stopped early to avoid overfitting on the image classification dataset. Weights were optimized with 0.9 momentum with an initial learning rate of 0.01, exponentially decaying with a factor of 0.95.

Even though the pre-trained weights should intuitively have been useful to the agent, they actually harmed extrinsic training. Perhaps the perception module overfit on the image classification task and the weights were too difficult to adjust for the Reinforcement Learning setting. While the agent initially appeared to be learning quicker during the early phases

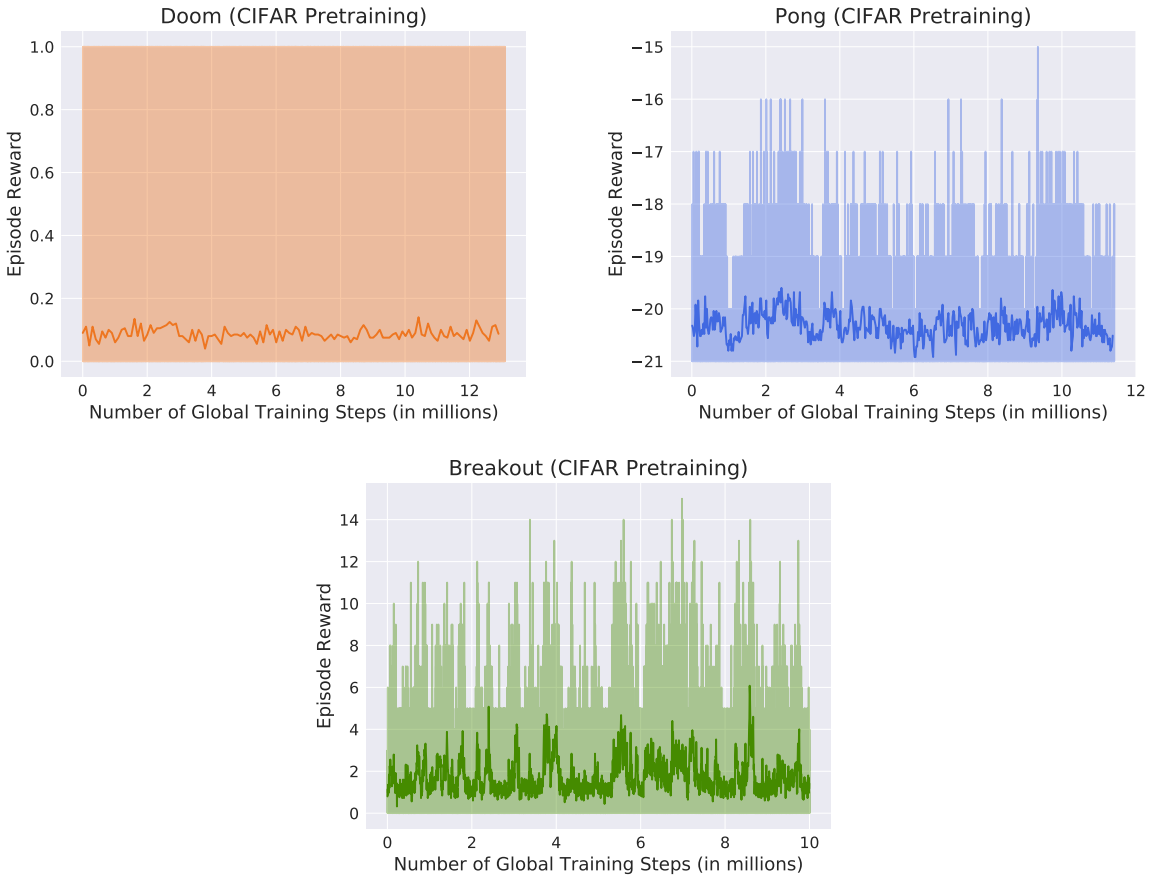

Figure 3: **Perceptual pre-training did not improve the following extrinsic training.** Across all environments, pre-training on CIFAR failed to improve the training speed of agents. In *VizDoom*, the agent made it to low performance levels. *Montezuma's Revenge* did not finish training in time.

of training for *VizDoom*, it ultimately did not achieve superior performance over the not pre-trained *VizDoom* agent. These results are quite counter-intuitive (and not conclusive due to time and resource constraints): simpler environments such as *Pong* and *Breakout* fail to train altogether after the perception module has been pre-trained. More complex environments such as *VizDoom* on the other hand may benefit from perceptual pre-training at first but still fail to train fully.

5.3 Curiosity-driven pre-training

The agents were pre-trained using the same configuration and hyperparameter settings as in the extrinsic case (Section 5.1), but with zero external rewards and with the additional intrinsically-based loss function as described in section 3.2, using $\beta = 0.2$. Figure 4 shows the training curves on curiosity alone: agents begin to learn about their environment quickly and decay after reaching a peak, seemingly because there is nothing new to learn, and they have a difficult time challenging their models of the environment. In other words, they get

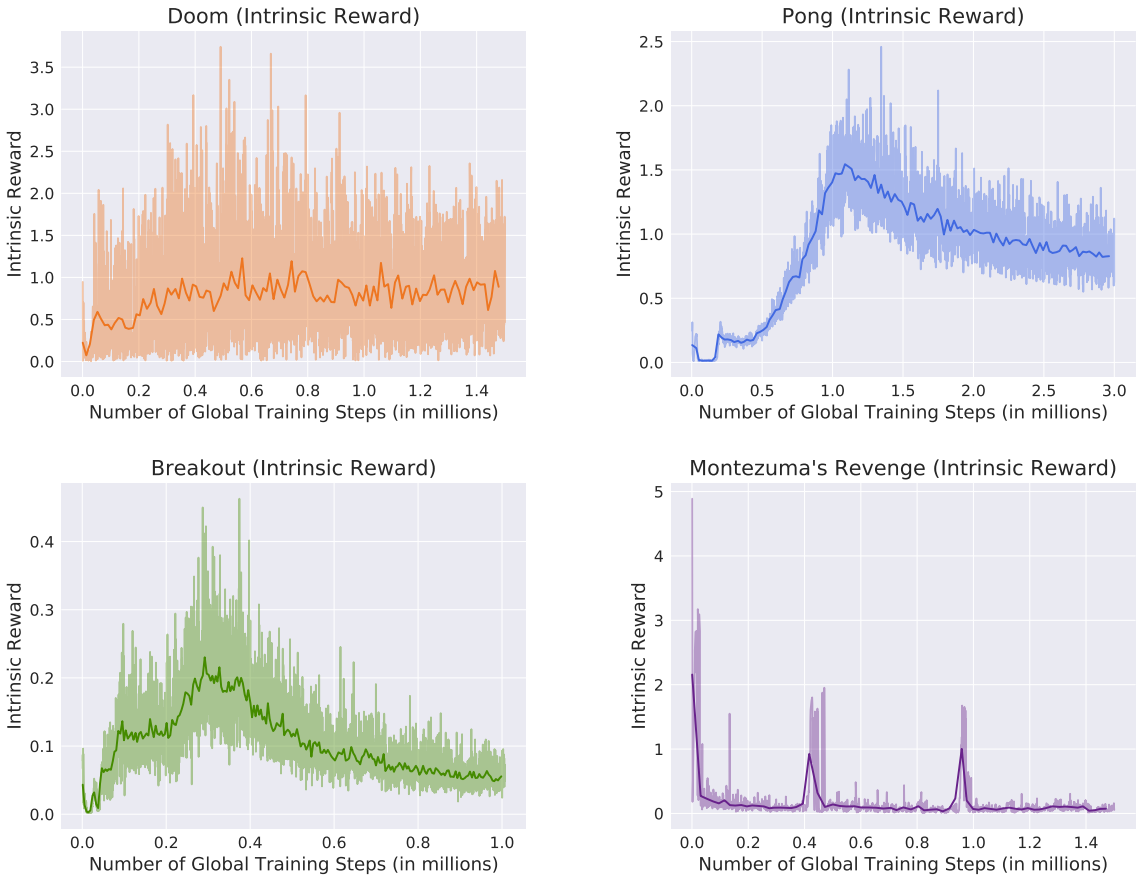

Figure 4: **Networks were pre-trained with intrinsic rewards.** Networks were initially trained on the respective environments with intrinsic curiosity rewards only. Then, after the pretraining, they were trained using extrinsic rewards only.

bored.

After the pre-training period, agents were fine-tuned by training with purely extrinsic rewards in the same manner as in Section 5.1, as shown in Figure 5.

The *VizDoom* agent, which was first pre-trained for 1.5 million steps, exhibits very slow improvement until approximately 7 million training steps in, at which point its performance increases rapidly until it plateaus near a reward of 1. This level of reward corresponds to the ability to find the vest on every run. Here, curiosity pre-training seems to exhibit a clear benefit, as performance is vastly superior to the not pre-trained and perceptually pre-trained cases. Thus, it appears that the curiosity-based intrinsic reward is teaching the agent something beyond simple image-recognition features.

With *Pong*, the agent that has been pre-trained for 0.5 million steps manages to achieve near optimal performance as in the extrinsically trained case. However, the agent takes significantly longer to achieve this performance, requiring near 7 million steps, whereas the agent that has not been pre-trained requires only 4.5 million steps. Furthermore, in our experiments, *Pong* agents that have been pre-trained for longer amounts of time (1 million steps and higher) fail to train at all within 10 million steps. This suggests that pre-training

may be harmful in this case, perhaps bringing the agent into undesired minima that are difficult to escape. Unlike with *VizDoom*, the visual setting may be too simple for curiosity to be of any use.

In the case of *Breakout*, we pre-train the agent for 1 million steps before fine-tuning. The curiosity pre-training appears to benefit the agent early on, as it quickly learns to achieve a reward of 5. While the agent steadily improves its performance thereafter, as in the pure extrinsic experiment, it ultimately performs worse, reaching a reward of approximately 13 (instead of 17) after 10 million training steps.

In *Montezuma's Revenge*, we pre-train the agent for 1.5 million steps before fine-tuning. The challenge in this environment is to avoid an enemy, find a key (reward: 100) and unlock a door (reward: 300). As visible in the plot, the agent is able to do this occasionally (whereas the agent that has not been pre-trained only ever finds the key), but is unable to learn a policy that allows it to reproduce this success consistently. This may simply be because the game is too challenging, perhaps more so than *VizDoom*. This requires not only navigation (as in *VizDoom*) but also the avoidance of an enemy and a very specific set of movements.

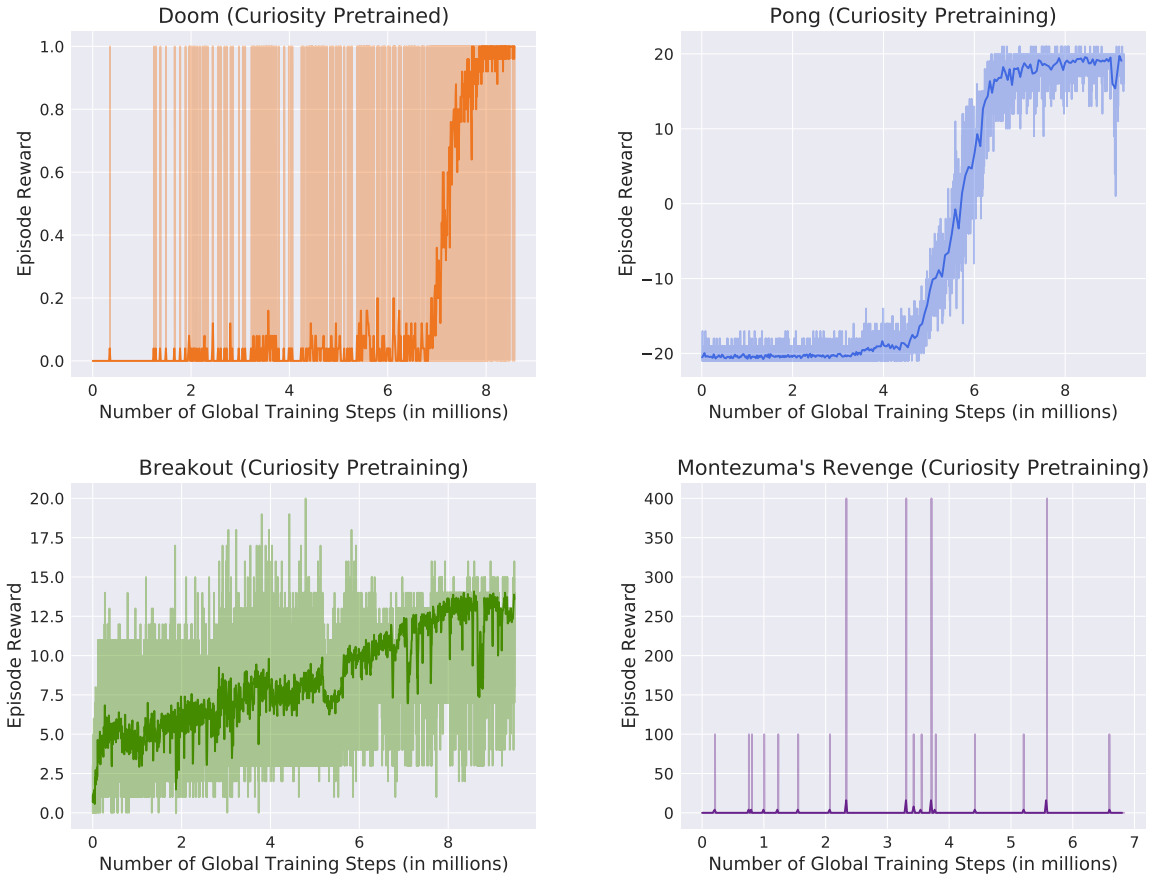

Figure 5: **Networks pre-trained with curiosity improved extrinsic fine-tuning only for some games.** For the most visually complex environment *VizDoom*, curiosity pre-training helped the agent achieve superior performance; however, for the simpler environments, it harmed extrinsic training (cf. Figure 2).

5.4 Comparison

We aggregate results from the previous sections and analyze successful training as well as training time until max performance. Table 1 shows the threshold of maximum performance for each environment and how many epochs it took each agent to get there. Many

	max	pre-training		
Environment	performance	none	perceptual	curiosity-driven
VizDoom	1	—	—	7.9
Pong	20	2.3	—	6.2
Breakout	10	3.2	—	5.7
Montezumas Revenge	400	—	—	—

Table 1: **Million epochs until maximum performance, with and without pre-training.** Many agents never achieve maximum performance. Pre-training only helps for VizDoom where curiosity-driven pre-training is the only technique that gets the agent to maximum performance. In simpler environments such as Pong and Breakout, pre-training harms extrinsic training.

of the agents fail to train to maximum performance within 10 million epochs. Without pre-training, 2 environments make it to maximum performance (Pong, Breakout). Perceptual pre-training seems to harm later extrinsic training and never makes it to maximum performance. Curiosity-driven pre-training additionally trains VizDoom to success but harms the training speed on Pong and Breakout compared to no pre-training.

6 Conclusion

We evaluated two potential pre-training methods for Reinforcement Learning on a range of environments: perceptual pre-training where only the perception module of an agent is pre-trained on an image classification task, and curiosity-driven pre-training where the full agent is pre-trained in the target environment but without extrinsic rewards. While both techniques show some promise, they fail to improve training on a majority of the environments tested here (*Pong, Breakout, Montezuma's Revenge*), even when extrinsic training alone works with the same set of hyper-parameters. However, extrinsic rewards and perceptual pre-training made it up to at least low performance levels and curiosity-driven pre-training later yielded maximum performances. As always with negative results in Machine Learning, it is hard to conclude whether or not pre-training has merit. Given that the same hyper-parameters often worked without pre-training, their choice does at least seem to be non-trivial. Ultimately, we see both techniques as potentially promising suggestions for future research with the need for thorough analyses across a range of settings.

References

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller (2013a). "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602.

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2015). "Continuous Control with Deep Reinforcement Learning". In: *arXiv preprint*. arXiv: 1509.02971.

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). "Curiosity-Driven Exploration by Self-Supervised Prediction". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 488–489.

Pathak, Deepak, Parsa Mahmoudieh, Michael Luo, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, Trevor Darrell, and Pulkit Agrawal (2018). "Zero-Shot Visual Imitation". In: *International Conference on Learning Representations (ICLR)*.

Gu, Shixiang, Ethan Holly, Timothy Lillicrap, and Sergey Levine (2016). "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates". In: *arXiv preprint* abs/1610.00633. arXiv: 1610.00633.

Olshausen, Bruno A. and David J. Field (1997). "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision Research* 37.23, pp. 3311–3325.

Kingma, Diederik P and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations (ICLR)*.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Networks". In: *Neural Information Processing Systems (NIPS)*.

Houthooft, Rein, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel (2016). "VIME: Variational Information Maximizing Exploration". In: *Neural Information Processing Systems (NIPS)*, pp. 1109–1117.

Klyubin, Alexander S, Daniel Polani, and Chrystopher L Nehaniv (2005). "Empowerment: A universal agent-centric measure of control". In: *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. Vol. 1. IEEE, pp. 128–135.

Mohamed, S. and D. Jimenez Rezende (2015). "Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning". In: *arXiv preprint*. arXiv: 1509.08731.

Bellemare, Marc G., Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos (2016). "Unifying Count-Based Exploration and Intrinsic Motivation". In: *CoRR* abs/1606.01868.

Tang, Haoran, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel (2016). "#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning". In: *CoRR* abs/1611.04717.

Haber, Nick, Damian Mrowca, Li Fei-Fei, and Daniel L. K. Yamins (2018). "Learning to Play with Intrinsically-Motivated Self-Aware Agents". In: *arXiv preprint*. arXiv: 1802.07442.

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *International Conference on Machine Learning (ICML)*.

Riedmiller, Martin (2005). "Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method". In: *Lecture Notes in Computer Science*. Vol. 3720. Springer, Berlin, Heidelberg, pp. 317–328.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013b). "Playing Atari with Deep Reinforcement Learning". In: *arXiv preprint*. arXiv: 1312.5602.

Van Hasselt, Hado, Arthur Guez, and David Silver (2016). "Deep Reinforcement Learning With Double Q-Learning". In: *AAAI Conference on Artificial Intelligence*, pp. 2094–2100.

Schulman, John, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel (2015a). "Trust Region Policy Optimization". In: *Journal of Machine Learning Research*.

Schaul, Tom, John Quan, Ioannis Antonoglou, and David Silver (2016). "Prioritized Experience Replay". In: *International Conference on Learning Representations (ICLR)*.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *arXiv preprint*. arXiv: 1511.07289.

Krizhevsky, Alex (2009). *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto. arXiv: arXiv:1011.1669v3.

Schmidhuber, Jürgen (2010). "Formal Theory of Creativity and Fun and Intrinsic Motivation Explains Science, Art, Music, Humor". In: *IEEE Transactions on Autonomous Mental Development* 2.3, pp. 230–247.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). "OpenAI Gym". In: *CoRR* abs/1606.01540.

Kempka, Michal, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski (2016). "ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning". In: *CoRR* abs/1605.02097.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980.

Schulman, John, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel (2015b). "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *CoRR* abs/1506.02438.