Generalizable Reinforcement Learning for Object Recognition

Bhairav Chidambaram, Connor Soohoo, Rohan Choudhury

July 15, 2019

Abstract

The issue of generalization and robustness has long been a concern with deep reinforcement learning, despite its tremendous success in domains like Atari, Go and simple control problems. As RL problems involve iteratively sampling from the same environment over and over, the policies learned by deep RL algorithms tend to be brittle and break down in environments even slightly different from the ones they are trained on, leading to the notion of them 'memorizing' a solution. Tackling this issue can be addressed from both an algorithmic and domain standpoint. We provide a benchmark task and provide baseline results. Our proposed task is based on the MNIST recognition problem, and allows easy evaluation of test performance due to a constantly changing environment. We find that baseline deep RL methods are able to handle this environment quite well, and hope to provide an interface for researchers to evaluate the robustness of their algorithms in the future.

1 Introduction

Reinforcement learning (RL), especially deep reinforcement learning, has recently been one of the most popular topics of research in the artificial intelligence community. Common deep reinforcement learning methods methods such as constrained policy optimization [7, 5], deep Q-networks, and advantage actor critic (A2C) have allowed countless advances in the field. Major drivers of this progress include widely-used benchmark domains such as the Atari Learning Environment [1] and MuJoCo [2], both of which present tasks on which RL algorithms can be tested.

In addition to simple domains like games, reinforcement learning has been applied to various real-world tasks, especially in control and robotics [9] . As such, the ability of reinforcement learning methods to adapt to diverse and changing environments is crucial for its success in these areas. However, due to the very nature of reinforcement learning, which is based on repeatedly learning from the same environment, these methods have led to a somewhat overreliance on these domains: policies trained with highly complex algorithms on simple tasks generalize poorly [10]. For example, simply changing the background color in an Atari simulator resulted in a policy completely breaking down. Serious brittleness in the policies resulting from standard RL algorithms suggests either that the current algorithms are not sufficiently robust or that the simulators are not diverse enough to induce such behaviors.

Significant work has been done on the topic of generalization in deep reinforcement learning, and the topic is under active investigation. However, each work uses different sets of environments and benchmarks. For example, Kansky et al. (2017) propose a graphical model approach to the Atari Breakout environment, where the positions of obstacles can change. [13] Finn et al. (2017) train a policy that can be adapted to new test environments, and Duan et al (2016) develop a policy that automatically adapts to environment dynamics.

On the other hand, the issue of overfitting and generalizability has been analyzed carefully in the context of supervised learning, a domain often considered somewhat orthogonal to reinforcement learning. In supervised learning problems, we are presented with a training dataset $\mathcal{X}_{train} \subset \mathcal{X} = \{x_1, x_2, ...x_n\}$, with fixed labels $\mathcal{Y}_{train} = \{y_1, y_2, ...y_n\}$. The supervised learning problem involves learning a map $f : \mathcal{X} \to \mathcal{Y}$ such that the error of f is as low as possible on test examples. Even in supervised learning, deep learning

techniques can have a tendency to overfit to training data [3, 8], implying that deep reinforcement learning techniques can be even more susceptible to this problem.

While one approach to tackling the generalizability is to introduce more complex simulators and algorithms, as previous works have done,a possibly simpler approach is to introduce common simulators and benchmarks that encourage *generalizable* behaviors, as is proposed in Zhang et al, and Packer et al. [3, 6]. Unlike typical reinforcement learning domains, in which an agent always acts in the same environment, introducing noise into the learning environment itself can be a simple way to encourage generalizable, robust policies. Providing an easy way to test the performance of a deep RL policy on a new environment would greatly benefit researchers interested in pursuing this direction. Zhang et. al develops three such examples, based on the two domains of image recognition and Atari [3]. In this project, we provide an example of such a domain and test common reinforcement learning benchmarks along with variants on it. Specifically, we use the classic MNIST task We hope to produce an example of reliable benchmarks for generalizable reinforcement learning and begin work to develop methods to encourage robust reinforcement learning policies and agents.

2 Background and Approach

2.1 Benchmark RL Methods

2.1.1 Proximal Policy Optimization (PPO)

Proximal policy optimization [5] is a simplified version of Trust Region Policy Optimization [7], in that it provides an unconstrained objective function that provides improvement across the board for differentiable RL algorithms. One of the challenges with training RL models with SGD is that an update might cause an unexpectedly large shift in the action distribution. PPO achieves the same performance as TRPO while solving this issue by simply clipping the objective so it does not change drastically during a single update. In practice, PPO is far simpler to implement and is just as effective as TRPO, and has proven to be successful across a wide variety of domains.

2.1.2 Deep Q-Networks (DQN)

The goal of Q-learning is to learn the true function from state-action pairs to reward, often referred to as the value function. When the number of such pairs is tractable, we typically use a table to keep track of the reward given every possible state-action pair. In cases where the state-space is extremely high dimensional, as is the case for our MNIST-GridWorld environment, tabular methods are less suitable; instead, we use a function-approximator such as a deep neural net as a stand-in for this table. This approach has proven effective in environments where game-state is encoded visually (e.g. Atari 2600 games) [12]. In these cases, convolutions are able to distill raw pixel data into a more meaningful representation of game state, and deep versions of classic Q-learning techniques achieve high reward.

2.1.3 Advantage Actor Critic

The Advantage Actor Critic, or A2C, algorithm uses an updating scheme that operates on fixed-length time steps of experience [4]. To improve training time we use an asynchronous version of A2C, which involves having multiple distributed worker agents trained in parallel, each with their own copy of the model and environment. Each worker computes an advantage function in addition to a reward function to assess how well a worker's actions fared over time. In other words, the advantage function is a metric to judge the long-term results of a specific action.

2.2 Approach

We formulated the classic MNIST image recognition task as a GridWorld reinforcement learning problem. Each grid-cell is a 7×7 square of the image (28×28), so that the agents moves on a 4×4 grid. The setup is as follows. Consider some image $d \in D$, where D is the MNIST training dataset. Our reinforcement

learning agent then begins at the center of the image $s_0 = (2, 2)$, and every grid-cell is initially blacked out. At every timestep t, the agent can choose to reveal either the grid-cell directly above, below, left or right of its current position s_t. Once we can reliably recognize the number in the image from the revealed cells, we end the game. For each additional time-step taken, the agent is given a reward of -0.1. If the agent correctly guesses a digit on a time-step, we provide an additional reward of +1.0. Note that at each time-step, the agent is both deciding which direction to move in and making a digit guess.

This encourages *generalizable* behavior because in every instance of the game, the underlying image d is different; the actions required to reveal the image will be significantly different. As such, achieving high reward in this task will require learning a policy that can succeed across all different classes in MNIST, with numbers in different orientations, styles, and thickness.

To further study the generalization ability of the agent, we introduced additional augmentation to the observation matrix. First we divided the pixel intensities by 255, constraining each entry in the observation to the interval [0, 1]. Then the two augmentation methods used were Noise, Normally distributed with mean 0 and standard deviation 0.1, and background intensity substitution, where the new intensity was distributed as Uniform(0, 0.5).

2.3 Implementation Details

We wrote a custom MNIST framework in OpenAI Gym [11] to be able to run basic image classification games. As opposed to previous work, we started a game at the center of an image as opposed to at some random pixel within the image. The below graphic depicts a given state of a sample playthrough of our MNIST game, where around half a dozen squares and part of the digit are currently illuminated.

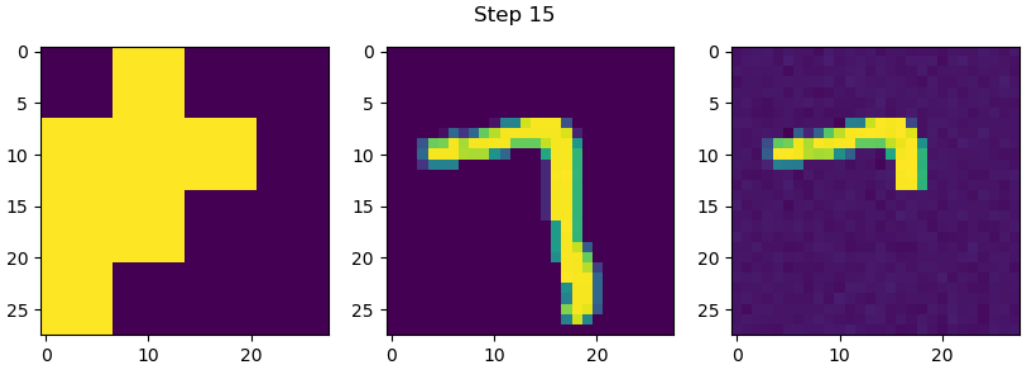

Figure 1: Sample play through of the custom MNIST game. The agent is currently at Step 15 of the playthrough, with part of the digit illuminated. The agent will then elect to make a directional move to illuminate another square in addition to a digit guess. Implemented using OpenAI Gym.

We chose to use Shangtong Zhang's Deep RL library on GitHub [6], which provided modularized implementations of baseline RL algorithms in PyTorch. It provided further ease of implementation with its native integration with OpenAI Gym and its baseline algorithm suite. To train our algorithms, we used a discount factor of 0.8 and a learning rate of 10^{-3}. All algorithms were trained for 10^6 steps. For PPO and A2C, we also added entropy regularization with $\lambda = 0.01$. In order to configure our custom MNIST environment properly with the library, we flattened our action space to change it from `MultiDiscrete` to `Discrete` Originally, our custom MNIST environment's action space consisted of two actions: a move (up, down, left, or right) and a guess (10 possible guesses). We flattened this 2-tuple `MultiDiscrete` action space to its `Discrete` equivalent with 40 possible actions (combining 4 possible moves and 10 possible guesses). This gave us the foundation upon which we could assess these RL algorithms on the MNIST game formulation and test for further generalization.

3 Results

As depicted in the below figure, we ran our baseline RL algorithms of PPO, A2C, and DQN on our custom MNIST framework for OpenAI Gym. We computed the average episode training reward over time for each algorithm by averaging the rewards over 5 randomly seeded runs. A smoothed version of our results of the baseline algorithms on MNIST is plotted below. Note that in our game formulation, the maximum possible episode reward a game can achieve is 0.9.

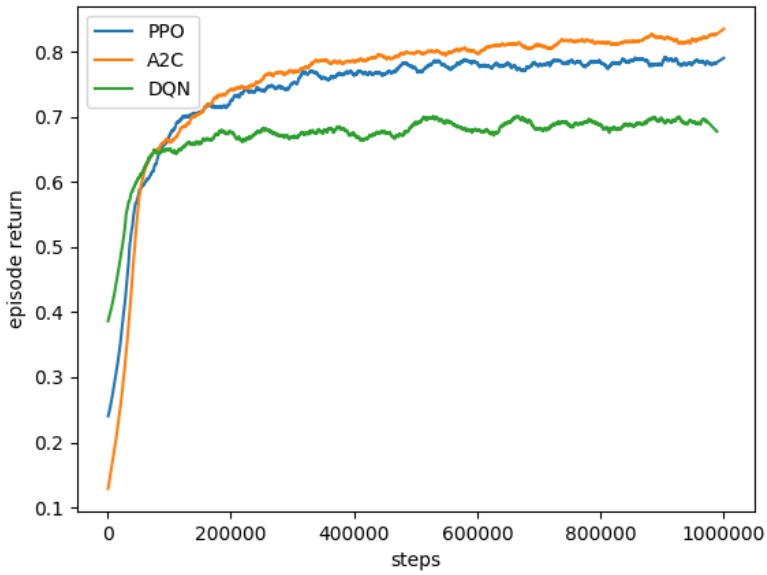

Figure 2: Episode training rewards for the RL baseline algorithms PPO, A2C, and DQN on custom MNIST Gym environment. Reward values of each algorithm for each time step were averaged over 5 randomly seeded runs and then smoothed with a Savitzky-Golay filter using a filter size of 10,000 and a polynomial degree order of 2.

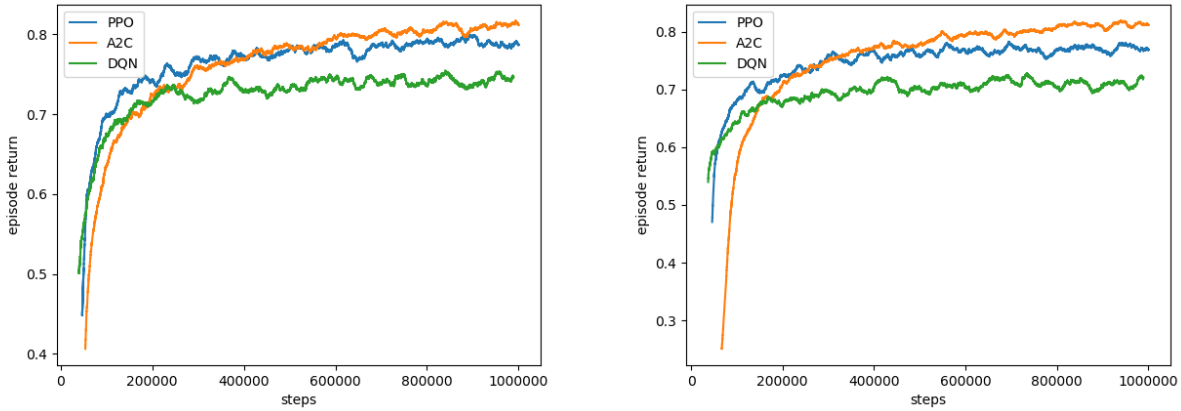

Figure 3: Episode training rewards for the RL baseline algorithms PPO, A2C, and DQN on custom MNIST Gym environment with augmentation. Uses same smoothing as in figure 1. On the left is the training curve with noise augmentation. On the right is training curve with background intensity augmentation.

4 Discussion

Based on our results, we have demonstrated the ability of A2C, PPO, and DQN reinforcement learning algorithms to generalize. Furthermore, adding substantial noise and background intensity augmentation did not significantly compromise the generalization performance of any of these algorithms. The most interesting result is the slight improvement in performance of DQN with noise augmentation compared to no augmentation. We hypothesize that this boost in performance is a result of similar advantages provided by data augmentation in the image classification domain. DQN is especially well-suited for RL environments with image-encoded state (e.g. Atari) so it would be interesting to see if these augmentation techniques can actually improve results in other more popular RL environments.

The key takeaways from our results are that (1) RL has the ability to generalize and (2) RL may benefit from data augmentation. Future work would involve investigating hypothesis 2. Traditionally, RL tends to use regularization in the form of forced exploration (e.g. an entropy term in the loss), so the only form of regularization is in the action space. Our work suggests that regularization to the state space in the form of data augmentation may also be effective in improving performance of RL algorithms. In many environments, the state changes minimally along a single trajectory, making it difficult to efficiently sample from a wide distribution of states (current state of the art RL solves this problem by throwing more computational power). Our hypothesis is that data augmentation to the states may better allow the agent to generalize to new states, making RL more sample-efficient.

References

[1] Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. *The arcade learning environment: An evaluation platform for general agents*. Journal of Artificial Intelligence Research 47:253–279, 2013.

[2] Todorov, E., Erez, T., and Tassa, Y. *Mujoco: A physics engine for model-based control*. In IROS, 5026–5033. IEEE, 2012.

[3] Zhang, A., Wu, Y., and Pineau, J. *Natural Environment Benchmarks for Reinforcement Learning*. Facebook AI Research, Nov 2018. https://arxiv.org/pdf/1811.06032.pdf.

[4] Volodymyr Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. *Asynchronous Methods for Deep Reinforcement Learning*. In ICLR, 2016. https://arxiv.org/pdf/1602.01783.pdf.

[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. *Proximal Policy Optimization Algorithms*. OpenAI, Aug 2017. https://arxiv.org/pdf/1707.06347.pdf.

[6] Shangtong, Z., *Modularized Implementation of Deep RL Algorithms in PyTorch*. GitHub, 2018. https://github.com/ShangtongZhang/DeepRL.

[7] Schulman, J., Levine, S., Moritz, P., Jordan, M.I. and Abbeel, P. *Trust Region Policy Optimization.* http://arxiv.org/abs/1502.05477

[8] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. *Understanding deep learning requires rethinking generalization. International Conference on Learning Representations, 2017.* https://arxiv.org/abs/1611.03530

[9] Haarnoja, T., Zhou, A., Ha, S., Tan, J., Tucker, G., and Levine, S. *Learning to Walk via Deep Reinforcement Learning.* arXiV Preprint. https://arxiv.org/abs/1812.11103

[10] Packer, C., Gao, K., Kos, Jernej., Krahenbuhl, P., Koltun, V., Song, D. *Assessing Generalization in Deep Reinforcement Learning.* https://arxiv.org/abs/1810.12282

[11] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. *OpenAI Gym.* arXiv e-print. https://arxiv.org/abs/1606.01540

[12] Mnih et al. *Human-level control through deep reinforcement learning. Nature, 2015*

[13] Kansky, K., Silver, T., Mely, D., Eldawy, M., Lazaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. *Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics.* arXiv e-print. https://arxiv.org/abs/1706.04317