Deep Reinforcement Learning with Actor Critic

Christopher M. Lamb
Department of Computer Science
University of California San Diego
San Diego, CA 92122
c2lamb@eng.ucsd.edu

Daniel Reznikov
Department of Computer Science
University of California San Diego
San Diego, CA 92122
drezniko@eng.ucsd.edu

Luke Liem
Department of Computer Science
University of California San Diego
San Diego, CA 92122
lliem@eng.ucsd.edu

Alexander Potapov
Department of Electrical and Computer Engineering
University of California San Diego
San Diego, CA 92122
apotapov@eng.ucsd.edu

Sean Kinzer
Department of Computer Science
University of California San Diego
San Diego, CA 92122
skinzer@eng.ucsd.edu

Christian Koguchi
Department of Electrical and Computer Engineering
University of California San Diego
San Diego, CA 92122
ckoguchi@eng.ucsd.edu

Abstract

Deep Reinforcement Learning (RL) is a powerful approach to agent learning in complex environments. Its applications range from gaming, to robotics to financial-market trading, achieving state-of-the art results across some distinctly challenging problem spaces. The aim of this work is survey the actor-critic based techniques that have been applied to agent learning in Atari games [1]. We profile baseline approaches, and experiment with changes to network topologies and configurations as well as the asynchronous multi-agent learning described by Minh et al [2].

1 Introduction

The actor-critic paradigm is simple and logically expressive. The actor's role is engage in some (pre-determined) policy, and the critic is responsible for evaluating this policy. Through interaction with the environment (in our case game-play) and explicit rewards (points scored) both the actor and critic are able iteratively update the representations of their tasks. Our goal is to profile a baseline approach which uses a single-agent advantage actor-critic model (where the actor is learning a policy through policy gradient [5] and the critic is learning a value function similar to classic Q-Learning [4]), and explore ways we can modify the network to achieve better performance. We also try to reimplement A3C [2] for asynchronous multi-agent play what has recently been shown to reduce learning time drastically. In sum, our efforts represent a survey of the state-of-the art methods,

which some experimentation do understand more deeply how network topologies (architectures and convolution vs linear approaches) affect both the ability of agents to learn and the training time.

2 Related Works

2.1 Methodology

2.2 System

Our group is lucky to have access to a GPU machine with 4 core Intel i7 processors, 32GB of RAM, and a powerful Nvidia GTX-Geforce 1080-Ti GPU with 11GB of VRAM. We install OpenAiGym [1] to support Atari game-play and PyTorch [3] for neural network (NN) development.

2.3 Experiment Setup

We clear Cuda GPU cache to avoid biasing from data cache hits in profiling run-times experiments.

2.4 Metrics

To compare the performance across experiments, we measure *score_vs_episodes*. To understand who the experiment configuration affect training time, we also report on *time_per_episode*. For a comparison, we thought it would be interesting to include the performance of human players as reported by Minh et al [2]. The human score is given by the 75th percentile score achieved by an average of human experts.

2.5 Games

In order to profile a wide-range of difficulties, we identify easy, medium and hard games. While the size of the action space varies from $[4, 20]$, the major differentiating factor for complexity comes from the delay in reward assignment. Pong and Breakout and the easy games, Space Invaders is the easy game, and Berzerk is the hard game. The following table uses a metric to quantitatively measure game complexity, the results are summarized:

Game Name	Difficulty Score	Category
Pong	2	Easy
Breakout	3	Easy
Space Invaders	5	Medium
Berzerk	9	Hard

Figure 1: Game Difficulties

OpenAIGym exposes two methods of game-play. RAM provides the memory state of the game, and RGB which only provides a RGB image of the screen, reflecting the pixel values that would be rendered for game-play. Each action provided by the agent is repeatedly performed for a duration of k frames where k is uniformly sampled from $\{2, 3, 4\}$. We exclusively use RGB mode, which provides our model with the same information that a human player would have.

3 Experiments

References

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[2] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.

[3] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[5] A. Karpathy, "(3) deep reinforcement learning: Pong from pixels."

[6] OpenAI, "Openai baselines: Acktr and a2c," Nov 2017.

[7] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *CoRR*, vol. abs/1710.02298, 2017.