Exploring the use of data preprocessing reinforcement learning to play Pong

Xingzhi Lu

Concord College

Abstract

1 Introduction

In recent years, there has been a boom in the use of computer vision models for reinforcement learning. However, as seen in other computer vision models, it can often be difficult for the model to extract the key information from the image. This is often done through training convolutional neural networks, which takes up significant computing memory and time. Therefore, the research aims to explore whether there is a more efficient alternative to CNN models through manual feature extraction.

2 Literature Review

2.1 Previous work

Since DeepMind developed the first deep reinforcement learning that automatically learns to play various Atari games in 2013 [1] with a Deep Q-learning Network (DQN), there has been many researches that adopts deep reinforcement learning, due to how it demonstrated the viability of training a completely computer-vision model that acts as an smart agent. It is worth noting that there were at least 30 papers published from 2014 to 2017 after DeepMind's breakthrough in DQN in 2013 [2].

Some of the key researches after 2013 include how Levine et al. trained a robot arm to complete manipulation tasks simply using a computer vision model, [3] and how Gu et al. extended deep reinforcement learning to a continuous action space in robot arm control.[4]

The use of self-playing in training was inspired by DeepMind's 2016 research on AlphaGo, which was the first Chinese Go playing AI that defeated a world champion in Go. [5] OpenAI subsequently trained a model that solves Rubik's Cube with a robot hand. [6] These researches showed how self-playing could solve the issue of having insufficient training data.

3 Methodology

In this investigation, we would look at how a model that takes in the last few frames of the gameplay directly performs as compared to a model that takes in some features extracted from the last few frames of the gameplay in various games. We would measure the performance of the models by comparing number of epochs used to reach a target level of reward.

3.1 Games considered

In this experiment, I will consider the following games listed. Those games are either single-player or the agent is being played against an AI.

3.1.1 Pong

I have used a simplified version of Pong, removing the more realistic physical simulations, so that the game consists solely of reflections of the ball when hitting the paddles or the boundaries. The game is rendered in 0s and 1s using a 160×210 array in computer vision mode, and returns the current position, speed of the balls and the paddles when using feature extractions. To make the reward less sparse, I have altered the reward function so that there is reward for each time the agent hits the ball with the paddle.

3.1.2 Flappy bird

I used the external libraries available on GitHub. The rewards are: +0.1 for every frame it stays alive; +1.0 for successfully passing a pipe; -1.0 for dying; −0.5 for touching the top of the screen. [7], [8]

3.1.3 Space invaders

I used the external library ALE_PY.

3.1.4 Pac-man

I used the external library ALE_PY.

3.2 Control variables

Both models used will have a roughly equal number of learnable parameters and are all trained under the same learning rate and other hyperparameters.

3.3 Model architectures

3.3.1 Multi-frame computer vision model

The model includes 3 convolutions with ReLU in between, followed by 3 fully connected layers, and each layer has a ReLU function after it as the activation function.

3.3.2 Feature extraction model

The model includes 3 fully connected layers, and each layer has a ReLU function after it as the activation function.

4 Evaluation of various gaming methods

4.1 Multi-frame computer vision model

Game	Target reward	Epoch needed to achieve

4.2 Feature extraction model

Game	Target reward	Epoch needed to achieve

5 Limitations and future work

Due to the limited availability of computational resources (only one A100 GPU on Google Colab), I could not test out models with a larger number of trainable parameters due to the shortage of memory and the unacceptable long training time. The research was solely done on video games where feature extraction is straightforward, and the game play is not complex. Therefore, this research might not apply to the more complex games where the current game state cannot be solely represented via a tensor consisting of a few numbers. Furthermore, the games all have straightforward reward function designs, and the reward functions of the games were specifically designed to ease the training process and to make the rewards less sparse. In games where there reward of the game is sparse or difficult to compute, such as when the target of the game was to accomplish specific target tasks, the reinforcement learning method might fail to learn to play the game effectively.

6 Conclusion

...

Bibliography