Deep Reinforcement Learning and Transfer Learning with FlappyBird

Cedrick Argueta, Austin Chow, Cristian Lomeli
Department of Computer Science
Stanford University
Stanford, CA 94305
{cedrick, archow, clomeli}@cs.stanford.edu

Abstract

Reinforcement learning's growth in popularity in recent years is partly due to its ability to play some video games with a level of mastery that no human can reach. Transfer learning is popular in the field of deep learning, and using pre-trained models on certain tasks speeds up training time and increases performance significantly. In this project we aim to apply transfer learning to the popular video game *FlappyBird* and analyze its performance compared to traditional reinforcement learning algorithms.

1 Introduction

Reinforcement learning is a technique for solving certain decision tasks, where an agent attempts to maximize a long-term reward through trial and error. This trial and error paradigm is better explained through the lens of exploration and exploitation: the agent must decide whether it explores and learns new information, or exploits the current policy to maximize reward. We will primarily explore the applicability of deep Q-learning, a deep reinforcement learning algorithm, to the FlappyBird game. In addition, we intend to explore the impact of transfer learning, a machine learning technique where a model developed for a specific task is reused as the starting point for a model in a second task. Being able to transfer learn will allow us to develop a more general model for different tasks, saving time and effort. In particular, we aim to see whether transfer learning will speed up training time to reach the same performance level as the baseline, or whether transfer learning will allow the agent to reach even higher performance levels than the baseline.

2 Related Works

The impetus behind our project is DeepMind's paper, "Playing Atari with Deep Reinforcement Learning". Mnih et al. were able to show how a Deep Q-Network could take raw image pixels from an Atari game and estimate the Q function. [3]

Their model also takes advantage of recent advances in convolutional neural networks for image processing, a target network for stabilizing policy updates, and experience replay for a more efficient use of previous experience. Transfer learning is the idea that generalization occurs both within tasks and across tasks. [7] Deep reinforcement learning may benefit from transfer learning, especially since convolutional neural networks are often used for playing games. Since convolutional neural networks often learn similar features in the first, second, etc. layers across a range of image classification tasks, it's possible that transfer learning in the context of reinforcement learning for video games can exploit this.

Preprint. Work in progress.

3 Game Mechanics

The game of FlappyBird can be described as follows: a bird flies at a constant horizontal velocity v_x and a variable veritcal velocity v_y. The bird can flap its wings, giving it a boost in altitude and velocity v_y. The aim of the game is to avoid randomly generated pipes that restrict the flying area, leaving only a small gap for the bird to fly through.

We model the problem as a Markov decision process with no knowledge of the transition probabilities or reward function at every state. The transition probabilities are unknown, since each state consists of the deterministic bird position and velocity, along with the non-deterministic pipe positions. The only reward signal received from the game in its standard implementation is when the bird flies past a pipe, giving us a reward of 1. This sparse reward makes it impossible for us to get an explicit reward for each (s, a, s') tuple. The start state s_{start} is the bird at some constant height, with no pipes on the screen. The actions for every state are the same: the bird can flap its wings or not. The only exception to this are the end states s_{end}, where the bird collides with a pipe or the ground. In these cases, there are no actions to take, and the episode ends.

4 Approach

The goal of our project is twofold: we aim to evaluate deep reinforcement learning algorithms on the FlappyBird game, and also experiment with transfer learning to analyze the impact it makes on the training process.

Vanilla learning methods like Q-learning are not well suited to this problem, since the state space of the game is very large. The position of the bird and pipes are continuous values, and as such we have an almost-zero probability of reaching a particular state that we've seen before. Thus, it will be necessary to use either function approximation to generalize to unseen states or some form of deep learning that can extract features automatically. We envision using Deep Q-networks as a nice foray into deep learning, given our experience with Q-learning.

Furthermore, we will demonstrate the ability for policies learned through deep reinforcement learning on other games to transfer to FlappyBird. The games on OpenAI's gym platform will interface well with the rest of our project. [2] [1] This has been demonstrated before, particularly through the use of convolutional neural networks for playing games directly from pixels. [3]

4.1 Input / Output

For the first part of our project, our model is a neural network that takes as input the game state in vector form and outputs an action to take. This neural network effectively approximates our Q-value function. The game state vector contains the bird position and velocity, as well as the pipe positions for the two next pipes. Outputs for the neural network are our actions, either a jump action or NOOP action. The model could be extended to take raw pixels as input, which would then increase the complexity of our model, as it'd require the use of convolution neural networks rather than the simple dense layers we have now.

Transfer learning will involve training a similar model on a different game, such as *Breakout* or *Pong*. Then this trained model would serve as the starting point for the FlappyBird model.

4.2 Infrastructure

The infrastructure for the game comes mostly from the PyGame Learning Environment, OpenAI gym, and keras-rl packages. The PyGame Learning Environment provides a nicely wrapped implementation of FlappyBird, complete with sprites and the relevant game mechanics built in. [6] Keras-rl provides a deep reinforcement learning framework that gives us a simple interface for training agents. We take advantage of these two here, writing simple wrappers for the PLE game instance so that it is compatible with keras-rl. [4] The keras-rl package additionally provides an implementation of several algorithms that are applicable to FlappyBird, particularly Deep Q-networks and the SARSA algorithm. Code for our project can be found at this link.

4.3 Baseline and Oracle

The baseline for this project is an agent trained with SARSA on an ϵ-greedy policy π_ϵ. The policy learned from SARSA algorithm described in Sutton and Barto updates the policy greedily based on Q_{π_ϵ}. [5] We see that Q_{π_ϵ} will rarely get past the first pipe, even after 500 training episodes. The policy performs similarly to a random policy, albeit it does maximize the reward that can be attained without passing the first pipe. The rewards structure for the baseline is as follows:

tick	0.1
passed pipe	1.0
collided	-10.0

where the agent receives a passive reward of 0.1 for being alive each tick, and additional rewards for passing pipes and colliding with objects.

Another baseline for this project is an inexperienced human player. We have consistently been able to reach 10 pipes, but still occasionally fail earlier.

The oracle for this project is the high score of a human who can play the game extremely well. Since the game isn't particularly difficult for humans and gameplay doesn't change dramatically throughout the game, it's possible for humans to play nearly indefinitely. After looking at high scores online through leaderboards and videos, the maximum legitimate score that we found was 999.

The gap between the baseline and oracle shows us that there is room for improvement, and deep Q-learning will be able to bridge that gap. When we have the model that bridges the gap, we can analyze the training time and model performance after different amounts of training, and compare that to models that are pretrained on other games like Breakout or Pong.

References

[1] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[4] Matthias Plappert. keras-rl, 2016.

[5] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2015.

[6] Norman Tasfi. Pygame learning environment. `https://github.com/ntasfi/PyGame-Learning-Environment`, 2016.

[7] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *JMLR*, 10, 2009.