Image Processing for Discovering the Rules of Video Games

Erik Culberson, Neil Ferman, and Viljo Wagner

1 Introduction

There are many examples of machine learning algorithms being able to play video games when the algorithms are integrated directly with the game. This is a costly process and needs hours of development work as well as needing more access to resources such as the source code of the video game. This approach also gives algorithms more in-game knowledge than a normal player would have. By removing this advantage from the algorithms we will be able to see how well the algorithms can compare to human players given an equal playing field. General Video Game Playing AI is a little newer, but has been approached in [2]. Given the same inputs as a human player, a screen image as an input and a set of key controls as output, can the algorithm discover the set of rules of the game as well as the rules of the game world?

2 Problem

Current research into artificial learning in games focus heavily on algorithms' ability to discover the optimal or fastest solution to the current level or problem. The algorithms are given the rules and knowledge of the game and the environment before hand. This forces the algorithm to only be able to be used for one specific game. To create a general algorithm that can play multiple games, the first step is to allow the algorithm to learn the rules of the game on its own.

3 Game

We chose to use Ms. Pac-Man as our test game for this project. We chose this game due to its simplistic nature in understanding the game by image, but complication in AI. Ms. Pac-Man is a game where you play as Ms. Pac-Man, a character inside of a maze. Your objective is to eat little dots called pellets inside of the maze. There are four ghosts within the maze who chase Ms. Pac-Man that you want to avoid. The four ghosts are different colors and behave in slightly different ways. What makes the AI of Ms. Pac-Man interesting is how the ghosts have an expected behavior, but also have a random chance of taking a random route instead of following their expected behavior [3]. This makes the AI less predictable for a player (and indeed for an AI) to learn.

4 Algorithm Structure

Several different algorithms were needed to achieve our goal. We will break each overall algorithm into steps for the project and go into detail of how each algorithm works. The steps are as follows: 1) **Image Capture**: Capture images of the game as it is played, frame by frame. 2) **Image Recognition**: Convert the image into an array of colors and coordinates, and separate images by parts. 3) **QLearning**: Pass the encoded images as well as a set of valid inputs into our QLearning algorithm, to be used to learn the rules of the game and update accordingly.

5 Image Capture

An image capture script written in Lua was used to interface with our platform to capture frames from the game. The image capture takes screen shots of the game every 10 frames. Since the game is being run at about 60 frames per second, it captures 6 frames per second.

6 Image Recognition

With each image captured, the next algorithm is to convert each image using opencv so it can be read and encoded by the QLearning algorithm.

7 QLearning

QLearning is an algorithm that uses Model-Based Learning. It is an evolved form of Markov Chains. To summarize QLearning, it takes a state, and then has transitions from its current state into future states. Each transition has a probability of being taken to move into the next state. Each possible state that can be transitioned into (known as a future state) has a reward or penalty associated with it. QLearning will at first pick a future state at random, and then, based on whether a reward or penalty is reached, update the transition probability. What makes QLearning so appealing for learning games is that it can on occasion, take a route that seems sub-optimal to explore whether that future state (or any subsequent future state) might lead to greater rewards. This is good for games that have random chance outcomes that are sometimes bad, but sometimes good.

8 Results

Given the same inputs as a human player, a screen image as an input and a set of key controls as an output, can the algorithm discover the set of rules of the game as well as the rules of the game world. Using an algorithm to read in the image the algorithm will press a key corresponding to the image. By doing this the process will create a feedback loop that allows the algorithm to see what that key does in the frames that precede it.

9 Conclusion

References

[1] S. Sabour, N. Frosst, and G. E. Hinton, Dynamic Routing Between Capsules, Neural Information Processing Systems, 2017.

[2] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. Lucas, "General Video Game AI: Competition, Challenges and Opportunities," Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[3] T. Thompson, "What's the Deal With Pac-Man?", Exploring artificial intelligence in games: in academia, indie and AAA gaming, 2014. Available: https://aiandgames.com/ai-and-pacman/

[4] S. Raval, "Q Learning Explained", December 1, 2017. Available: https://www.youtube.com/watch?v=aCEvtRtNO-M

[5] J. Bradberry, Introduction to Monte Carlo Tree Search, 2015. Available: https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/

[6] Harrison, Using a neural network to solve OpenAI's CartPole balancing environment, 2017. Available: https://pythonprogramming.net/openai-cartpole-neural-network-example-machine-learning-tutorial

[7] J. Levine, Monte Carlo Tree Search, 2017. Available: https://www.youtube.com/watch?v=UXW2yZndl7U

[8] K. Stanley, R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies", The MIT Press Journals, 2002. Available: http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf