Human Actor as Policy for a Reinforcement Learner During Learning

Johnathan Kunz

Abstract:

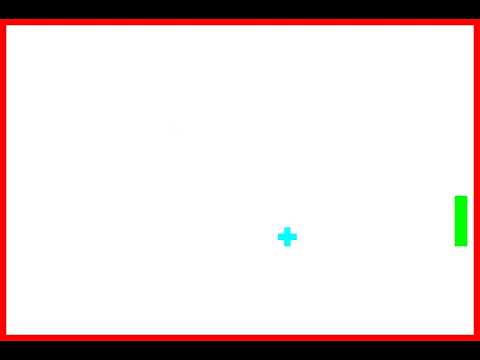
Game AI is an important piece of any video game. Good AI's in games like halo 1 imbue their games with incredible replay-ability and immense amounts of depth. Developing good AI's is a difficult task, often failed. Here, I experiment with training an agent to play a simple game I call bounce, a pong-like, where the task is to bounce a ball off a wall as many times as possible. The hope of this system is that an expert actor can train the agent to play the game quicker than the agent would learn on its own. This if successful would lead to an easier way to train AI’s and have them trained well.

Intro:

Game AI is an important piece of any game, and good Ais are hard to develop. When games have good AI they are much more fun to play. AI’s in games are usually hand coded to respond in certain ways to certain player actions, or to try and reach some predetermined goal. Neural Networks have started to be experimented as options for game Ais. These have recently been learning how to play simple Atari games, to good results. These Neural Nets take a relatively long time to learn how to play even these simple Atari games well. The goal of this paper is to combine the hand coded AI’s or expert players, with Neural Networks so that Neural Networks can learn to play a game better, and faster.

Methods:

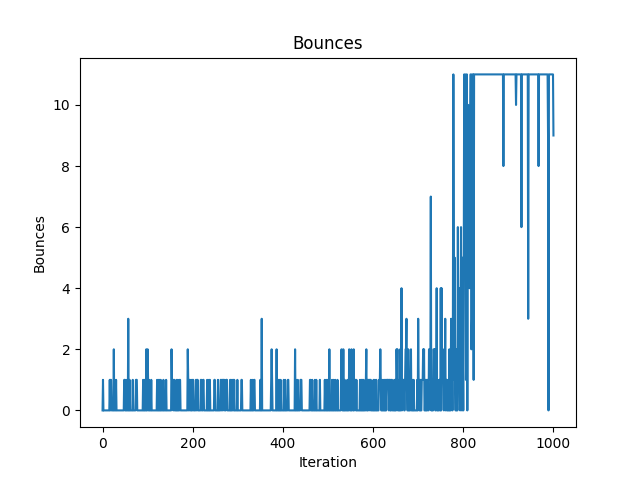
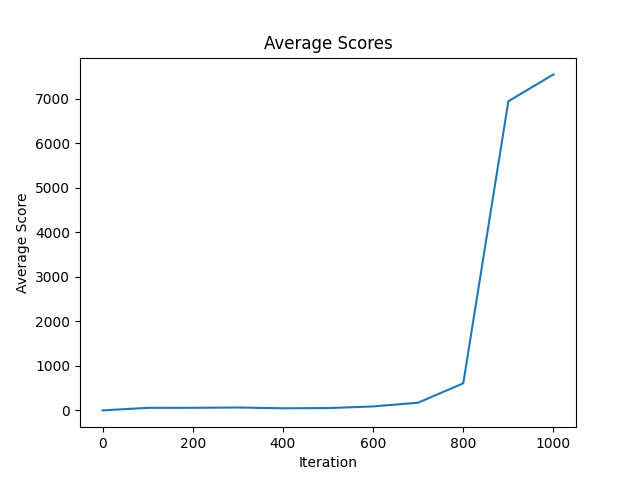
The system used in experiments is a custom gym environment, and a deep learning neural network. The deep learning neural network base code draws from the tutorials shown in class for one assignment 3, this is linked in the sources section as source number 2. The gym environment is a simple custom-built game, like pong, which I call bounce. The goal of the game is to bounce a ball off a wall as many times as possible (max of 10) without letting it hit the wall on their side of the field. In the video below, the ball is in teal, the paddle is in green, and the borders of the play field are in red.

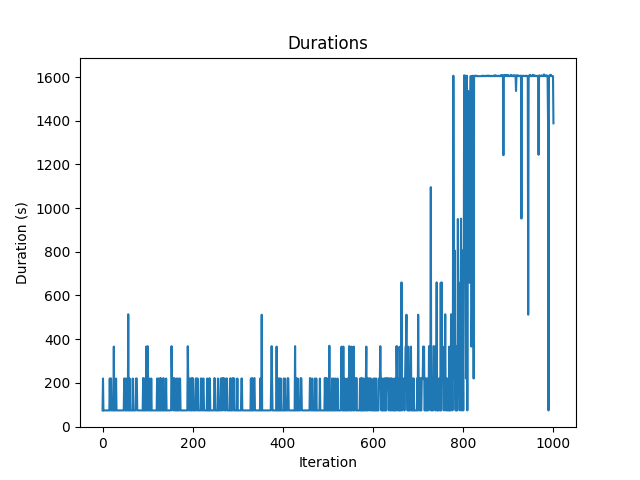
[](https://www.youtube.com/embed/Ml3bbmDXvvc?feature=oembed)

I ran experiments on the environment to get baselines about its solvability. I tried a simple Q-Learning algorithm before moving onto the deep learning neural network. After this, I ran experiments replacing the normal policy of the deep learner with one that get an expert actors action as a replacement for the policy and uses that to train the neural network. These experiments have 2 parts, the first where an expert adds experiences to memory, and the second where the agent learns from the experiences.

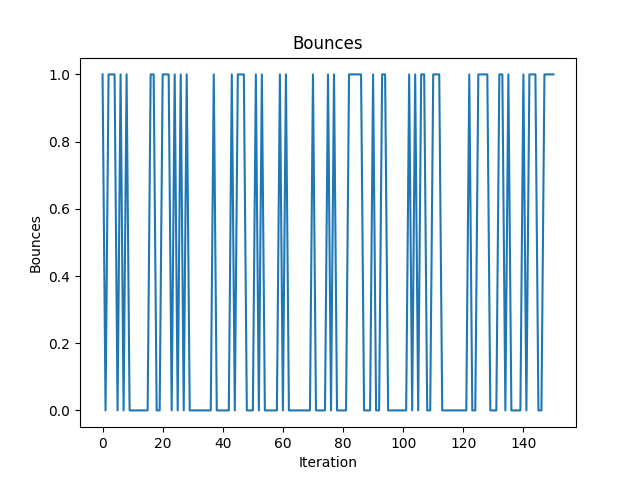
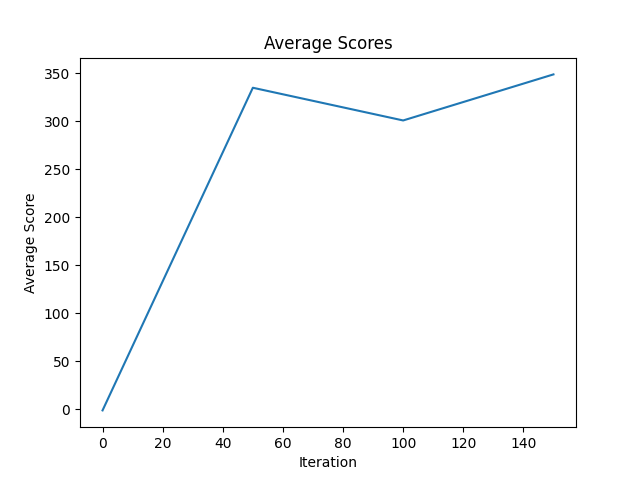
Results:

It was difficult to get the neural network trained on the system even without human involvement. I wanted to make sure the system was set up correctly and solvable before trying the more complex problem. I did eventually get the problem solving properly in about 1,000 iterations. The reward function I used is based on if the ball and the paddle are aligned, if they are the reward increases by 5, and if the game ends by the ball passing the paddle the reward is subtracted by 25. Here are the images from tests of the learner: the images plot the average score obtained, the duration of each iteration, the number of bounces made in each iteration, and score for each iteration. The most telling plot is the plot of bounces made, as that is the most telling indicator of the agents success.

Chart, histogram

Description automatically generated

After this I ran experiments where the deep learner attempts to learn from the experts actions. This was accomplished by collecting experience from the experts actions and then learning from them without adding anything new to memory. In these experiments the reward structure is 1 for every action the expert takes, and 0 for everything else. These experiments went poorly, the learner was not able to learn from the expert or learn to solve the problem. Here are the images from these experiments:



Chart, bar chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

I came to realize that the problem with this method was that the rewards in the experts experience were uniform, all 1’s, and the system never learned the good parts of the actions to emulate. This led me to reverting the reward structure back to the structure of the first experiment where if the ball and paddle are even the reward increases by 5, and if the ball gets past the paddle the reward is decreased by 25. This method didn’t solve the problem either since the experiences are still uniform due to having only the experts perfect actions.

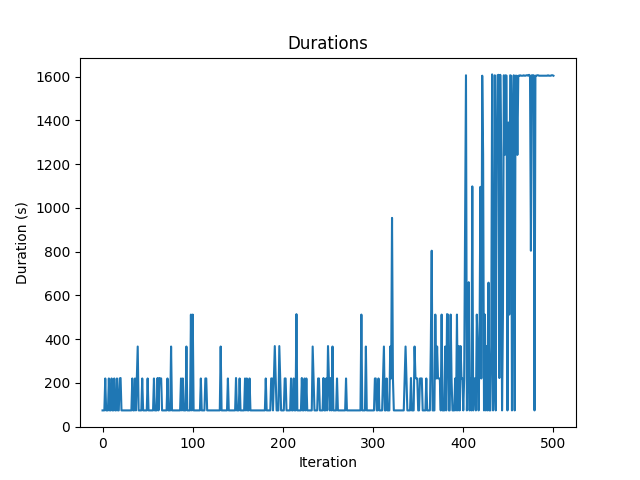
At this point I spoke with the Professor who advised modifying the experience gatherer so that it is like an epsilon learner, where it takes random actions according to epsilon, where epsilon decreases each step so that less random actions are taken as time goes on. This experiment also did not produce a actor that had learned to play the game. The problem here is even though the experience rewards isn’t completely flat, the learner still doesn’t have enough exploration.

For the next experiment the methods were modified a bit. During the first section the expert player collects experiences for the agent, during the second section the agent learns from the expert player while also exploring on its own like it normally would. This experiment proved successful, this agent learned to play the game well and learned this behavior in half the time it took the original agent to learn on its own. Here are the plots of this:

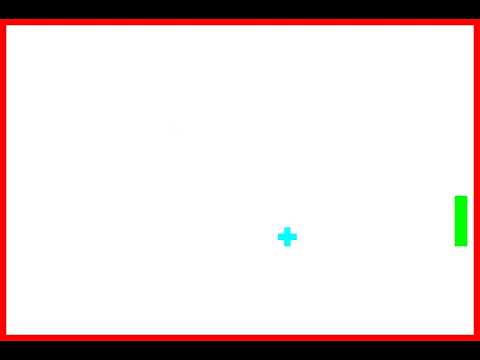
Chart, line chart

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Additionally, here is this agent in action, having learned how to play the game properly:

[](https://www.youtube.com/embed/Ml3bbmDXvvc?feature=oembed)

Summary:

The experiments ended with the agent learning how to play the game well twice as fast as it normally would have. It took many different experiments, and a lot was learned. The main problems faced were flat reward structures, and lack of proper exploration to learn how to play the game well. Flat reward structures caused the learner not to understand what the correlation between good and bad actions were and made the system unable to learn how to play the game well. This problem was accentuated by the lack of exploration, where the system didn’t know enough bad actions to be able to see that an action would be bad. These issues were resolved by having the experts experience gathering as a supplement to the standard deep learning system, where the agent gathers experience and learns as it goes. This combination allowed the system to have many examples of good actions, as well as enough exploration to understand why the good actions were good.

Conclusions:

Game AI is an important piece of any game, and good Ais are hard to develop. When games have good AI they are much more fun to play. AI’s in games are usually hand coded to respond in certain ways to certain player actions, or to try and reach some predetermined goal. Neural Networks have started to be experimented as options for game Ais. These have recently been learning how to play simple Atari games, to good results. These Neural Nets take a relatively long time to learn how to play even these simple Atari games well.

In this paper we discussed methods for creating AI’s using neural nets using an expert player to train nets faster and better than they normally learn on their own. Our results showed that this is possible at least in simple games. The next step would be to test training the AI on a more complicated game, probably a 3d game of some kind, or a more complicated puzzle style game.

Sources:

1. The code is held on github at: <https://github.com/PlipiKunz/RL-Project>, it is currently a private repository for the purposes of privacy while working on the project
2. The code base started from the Q-Learning with Images assignment which drew from this tutorial: <https://towardsdatascience.com/getting-an-ai-to-play-atari-pong-with-deep-reinforcement-learning-47b0c56e78ae>