**Ms. Pac-Man**

**A Methodological Study: Comparing Learning Algorithms**

Dylan Doelling  
 Computer Science  
 Utah State University  
 Logan, Utah, USA  
a02371420@usu.edu

Anthony Lasley  
 Computer Science  
 Utah State University  
 Logan, Utah, USA  
 a01644627@usu.edu

Abstract— Video games provide a consistent environment to test out different reinforcement learning algorithms. Ms. Pacman has been around for decades. There is a need to see which learning algorithms perform the best in this environment. While studying different reinforcement learning algorithms we are hoping to see if there is one that stands out. If there is, in the future, this can lead to more research as to why this one performed the best. Some of the algorithms we are looking to try are Prioritized Double Dueling DQN (PDD), and noisy N step PDD. We will be testing which algorithm stands out more.

*Keywords*—*Ms. Pacman, DQN, PDD, noisy N step PDD, reinforcement learning*

I. Introduction

Studying reinforcement learning is a challenge. There are a lot of unknowns when dealing with reinforcement learning algorithms and with a real-world environment. Using video games as a testing ground for reinforcement learning is a great way to study and develop reinforcement learning algorithms. Using Ms. Pac-Man as a controlled environment will help us to understand how different learning algorithms perform under a controlled environment. When testing the agents, we can witness the different strategies being performed or see which agent scores higher. The algorithms that will be tested are PDD, and noisy N step PDD. Both are variants of the DQN algorithm. Studying these algorithms on one environment will help with how each of these algorithms learn by the way they play.

II. Methodology

A. Brief Explanation of Models Used

PDD and noisy N step PDD are essentially more fancy versions of a deep Q network (DQN).

PDD: A simple explanation for PDD is that PDD will have a DQN that separately computes the value and action functions at the end of the network which will then combine for the final Q value. The way that PDD will choose its action is by an epsilon greedy strategy. PDD also uses prioritized memory replay.

A Normal DQN Neural Network Structure

A picture containing text, athletic game, sport

Description automatically generated[2]

A PDD Neural Network Structure

Diagram

Description automatically generated[2]

Noisy N step PDD: this model is basically the same as PDD expect for a few differences. One difference is that instead of having standard dense layers in the neural network at the end you use noisy dense layers which will add gaussian noise to its output. When choosing an action, it uses a greedy policy. This is because adding gaussian noise at the end of the neural network will ensure that the output already has some randomness to it. This will help the agent explore more.

B. Set Up Procedures

When training Ms. Pacman the control variables for each model are 0.0000625 for the learning rate, gamma at 0.99, mean absolute error loss function, maximum prioritized memory size of 1,000,000, and a runtime of 7 million steps. Or approximately 8,200 episodes. An episode consists of three lives (one in game round). These control variables will let us see which agent can get highest reward under the same conditions. The agent or Ms. Pacman will receive a reward of +1 for each item consumed. The items are dots, ghosts, and different kinds of fruits. For example, if Ms. Pacman eats 50 dots, 3 ghosts, and 1 fruit before she dies, her reward will be 54.

C. Experiment Procedures

There will be a set procedure to evaluate the model’s performance. For each model there will be 3 different checkpoints that will be tested. The three checkpoints we will be testing are at steps: 1 million, 6.5 million, and 7 million. A step is one action by an agent while testing. Overall, there will be 6 different experiments. At a checkpoint each agent will play for 50 rounds. A round is 3 lives. At the end of the 50 rounds there will be a score comparison. We will be comparing the average, median, high, and low scores of PDD and noisy N step PDD against each other at the given checkpoints.

III. Results

A. Overall Results

Chart, line chart

Description automatically generated

This learning curve graph is recorded up to the 8,230th episode. Noisy N step PDD during training. Exploration is on in both training and testing phases. We noticed that the initial learning was significantly higher than PDD. While PDD overtook noisy N step PDD at the very end. During the training phase we noticed that noisy N step PDD had a longer training time per episode than PDD. We suspect the cause to be the noisy layer of the neural network for the noisy N step PDD causing the agent to explore more.

B. In Game Video Comparisons at 7 million Steps

During training and testing we noticed that PDD was more consistent with its score. The average of PDD was higher than noisy N step PDD while noisy N step PDD both a higher high score, and a lower low score. We also noticed noisy N step PDD seemed to change up its behavior to avoid ghosts however, sometimes it did not work. An example of this would be that the agent would stutter when multiple ghosts were surrounding the agent. As soon as there was an opening the agent would make its move. PDD tended to just keep moving consistently. A consequence of this is that it will eat more ghosts. We are not sure if this is intentional or not. The PDD agent has never used the wrap around pipe in game. The wrap around pipe allows the agent to go from the right side of the screen instantly to the left side of the screen and vice versa. The noisy N step PDD tended to use the pipes more frequently. Overall noisy N step PDD has more variations to its behavior while the PDD agent tends to have a more set pattern with some variation.

C. PDD vs. Noisy N Step PDD at 1 million Steps

Chart, box and whisker chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | PDD | Noisy N Step PDD |
| High Reward | **76.00** | **118.00** |
| Median Reward | **50.00** | **97.50** |
| Average Reward | **50.70** | **94.22** |
| Low Reward | **40.00** | **58.00** |

The orange bar across the box is the median and the green triangle in the box is the mean. The circles are outlier scores while the other lines are one standard deviation from the median.

At this point in time noisy N step PDD is outperforming PDD by an average of 43.52 points (rewards). It seems like noisy N step PDD learns more quickly at the start. Noisy N step PDD also high a lot more variation in its scores compared to PDD.

D. PDD vs. Noisy N Step PDD at 6.5 million Steps

Chart, box and whisker chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | PDD | Noisy N Step PDD |
| High Reward | **147.00** | **148.00** |
| Median Reward | **138.00** | **141.00** |
| Average Reward | **138.96** | **136.88** |
| Low Reward | **137.00** | **105.00** |

At 6.5 million steps it seems like PDD close the gap from 1 million steps. PDD has a higher low and average reward compared to noisy N step PDD. However, noisy N Step PDD still has the highest score.

E. PDD vs. Noisy N Step PDD at 7 million Steps

Chart, box and whisker chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | PDD | Noisy N Step PDD |
| High Reward | **147.00** | **151.00** |
| Median Reward | **143.00** | **141.0** |
| Average Reward | **140.72** | **137.80** |
| Low Reward | **84.00** | **79.00** |

At the very end of PDD still has a higher average reward compared to noisy N step PDD. This time the gap is bigger from 6.5 million steps. Noisy N step PDD still has the highest score out of all testing. It is also interesting that both agents have an outlier that is reaped a very low reward.

F. Comparisons Between Same Models

Chart, line chart, box and whisker chart

Description automatically generated

The PDD agents have very low variance between their non outlier scores. At the 1 and 7 million steps there is more variance between the outliers compared to 6.5 million step agent. There is an upwards trend with the more training each agent was given.

Chart, box and whisker chart

Description automatically generated

Comparing noisy N step PDD agents against each other it seems that the longer you train it the higher the score will be. There is more variance in rewards compared to PDD. The average difference between 7 million and 6.5 million steps is a reward of 0.92.

IV. Summary

Overall noisy N step PDD tends to be more sporadic in its behavior, learning rate, and in game score. PDD tends to be more predictable in its behavior, less dramatic fluctuations in the learning rate, and less variation between in game scores.

V. Conclusion and Future Work

Noisy N step PDD learned more quickly during its initial training. At the end PDD had a higher recorded reward during training than noisy N step PDD. In testing noisy N step PDD had the highest reward. It was 151 compared to the highest PDD reward at 147. However, PDD had a better average reward during testing at 140.72 and noisy N step PDD average reward was 137.80. We think the reason that noisy N step PDD has more variation in its reward because of the gaussian noisy introduced into its neural network model. The neural network contains two fully connected layers with gaussian noise.

In the future we would like to continue training with noisy N step PDD and PDD to get to the point where they will successfully beat a level. We would also like to implement different learning algorithms.

V. REFERENCES

[1] J. Grigsby, “advancedpacmandqns: Extensions of Deep Q Learning,” GitHub. [Online]. Available: https://github.com/jakegrigsby/AdvancedPacmanDQNs. [Accessed: 16-Dec-2021].

[2] J. Grigsby, “Advanced DQNS: Playing pac-man with deep reinforcement learning,” Medium, 31-Oct-2018. [Online]. Available: https://towardsdatascience.com/advanced-dqns-playing-pac-man-with-deep-reinforcement-learning-3ffbd99e0814. [Accessed: 16-Dec-2021].

[3] “Keras-RL: Deep Reinforcement Learning for keras.,” GitHub. [Online]. Available: https://github.com/keras-rl/keras-rl. [Accessed: 16-Dec-2021].

[4] A. Lasley and D. Doelling, “Bigginlasley/Reinfocement\_learning\_ms\_pacman,” GitHub. [Online]. Available: https://github.com/bigginlasley/reinfocement\_learning\_ms\_pacman. [Accessed: 16-Dec-2021].