Improving the Performance of Deep Q-learning in Games Pong and Ms. Pacman

Kaiyuan Wu [1, *], Ningzhi Xu [2]

[1] Department of Bioengineering, Rice University, Houston, USA

[2] Department of Computer Science, Rice University, Houston, USA

* Corresponding author email: kvw1@rice.edu

Abstract. Unlike TD-gammon architecture, deep Q-learning algorithm combines deep learning and reinforcement learning, and it has achieved outstanding results in many Atari games. This study primarily focuses on two games, Pong and Ms. Pacman. It explores several approaches to improve the performance of deep Q-network (DQN). Based on the data obtained, while DQN displays a high-level performance in the simple Atari game Pong, it struggles a bit when learning the more complex game Ms. Pacman, leading to diverged loss. This under-performance may be partly explained by the shorter training time than the original paper due to limited computational resource. Given sufficient time of training and exploring, the model is believed to eventually converge once it identifies the optimal combination of hyperparameters.

Keywords: Deep Q-learning; Atari Game; Reward; Loss.

1. Introduction

TD-gammon, initially developed by Tesauro *et al*, is a backgammon-playing program based on reinforcement learning [1]. Unlike most previous neural networks, it is able to learn by self-playing. It primarily uses the multilayer perception (MLP) architecture, and over the years, it has achieved excellent results. However, later attempts of TD-gammon on other similar board games, such as GO, chess, and checker, are proven to be less impressive. This leads to the belief that TD-gammon only excels at playing backgammon, and its success is unlikely to be replicated in other games. To overcome the limitation of TD-gammon, this paper adopts the more recent approach of deep Q-learning, as outlined in Mnih *et al* [2], incorporating deep learning into the context of reinforcement learning.

Compared to traditional methods, DQN is more advantageous and efficient since it learns from randomized samples instead of the consecutive ones; thus, the correlation and variance between samples is reduced. Once the model is implemented, it is applied to two games, Pong and Ms. Pacman, to see how it performs in each case. Subsequently, the results are analyzed, and following attempts are made for improvement, followed by conclusions accordingly.

2. Method

2.1 Deep Q-learning

Unlike TD-Gammon, deep Q-learning incorporates a technique termed "experience replay", which is used to store agent's experiences before updating them based on random draws from the pooled data [3]. In essence, it enables the model to learn in small chunks each time, preventing the data distribution from being skewed. Subsequently, the algorithm proceeds with the epsilon-greedy policy for action selection and execution. After that, neural network's weights are updated according to the Bellman equation.

2.2 Implementation Details

The environments were set up using the Arcade learning environment [4] to include two games, Pong and Ms. Pacman. Agent's observations were provided in format of RGB image per frame for

each game, from which the agent would learn in order to better adjust its future actions. Specifically, in the game Pong, the agent played up to the total of 21 points per game, with a reward +1 being hitting the ball into the opponent's side and -1 being failing to catch the ball from the opponent. In Ms. Pacman, the agent received +10 points for each object it swallowed. The input to the DQN consisted of four consecutive images, each of which was carefully cropped to a size of 84 by 84 to fit the game setting. Concerning the architecture, two convolutional layers were incorporated into the model, with the first one having a kernel size of eight, stride of four, while the second one with a kernel size of four, stride of two. They were then followed by batch normalization. ReLU was also used to reduce the exponential growth in computation. Batch size was selected to be 64, and the model was trained for 1000 episodes in each trial.

3. Experiment

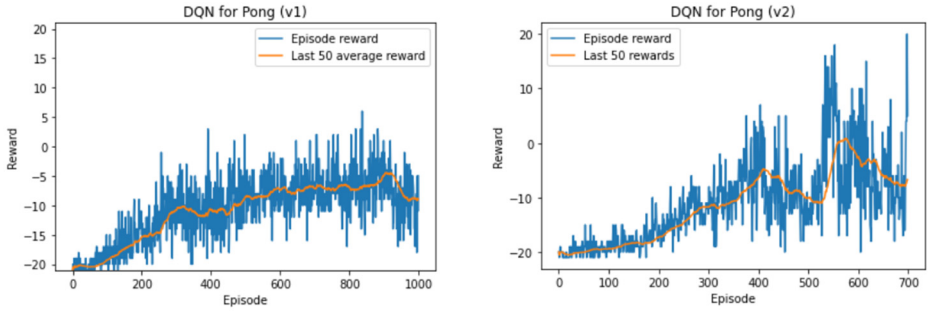
3.1 Pong

After training the DQN for Pong with a replay buffer size of 30000, the reward-episode graph was plotted below in figure 1(a). Overall speaking, the value of the reward gradually increased as expected; however, it seemed that the model encountered a plateau at around episode 900, where the average reward for the last 50 episodes reached a maximum value of approximately -5 and could not be improved further. We believed there were two possible explanations for that:

1. Since the implementation employed the epsilon-greedy algorithm with a shrinking epsilon value, at some point, the exploration may have ended too soon for the model.

2. The hyperparameter such as replay buffer size could be decreased, since Pong was a relatively simple game without much need for learning from old memories.

In this light, the model was updated for version 2 with a smaller replay buffer size of 20000, as well as a reduced rate of decreasing epsilon, to allow for more exploration. This time, the model performed much better, as indicated in figure 1(b). The model quickly reached an episodic reward over 10 rights after episode 550, despite the high variance. The last 50 rewards were also higher than the maximum moving average reward of -5 in the version-1 model. Due to computational constraint, the version-2 model failed to display rewards after episode 700, yet it was believed that if kept training for sufficient time, it would eventually converge to an optimal reward value of around 20, as suggested from the last spike at episode 700 in figure 1(b). Overall, the implementation of "Pong", as well as the general trend of episodic and moving average rewards, turned out to be roughly as expected.



(a) (b)

Fig 1. Reward-episode graph of DQN in Pong for (a) version-1 model, and (b) version-2 model.

3.2 Ms. Pacman

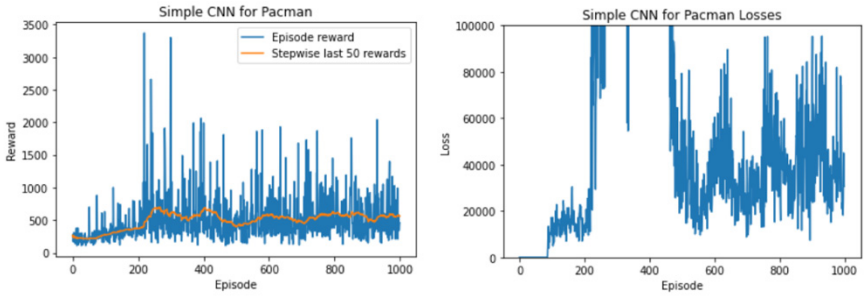
The second game, Ms. Pacman, turned out to be more difficult to train. For the original model, the reward graph in figure 2(a) showed that the moving average reward increased from 250 to 500, with a peak around 600 right after episode 200, but no further improvement was observed. In addition, from figure 2(b), the loss of training episodes was huge: it skyrocketed from 20,000 to over 100,000,

which made it difficult to converge. One possible reason could be the complicated nature of Ms. Pacman: its layout was labyrinthine, and the movements of enemies were thus unpredictable.

To address the issue, a deeper learning network was proposed. It consisted of three convolutional layers, followed by four linear layers, which intended to enable the model to learn more complicated behaviors. The results of this revised model with the deeper, more complex neural network were plotted below in figure 3. Unfortunately, the reward and loss graphs did not seem to show a significant improvement compared to the previous results in simple DQN.

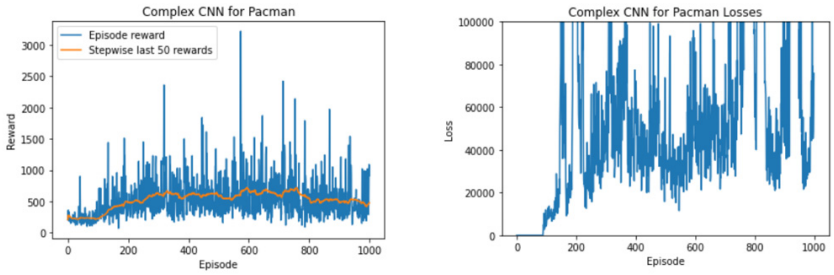
After careful analysis, we believed the problem may have been attributable to image processing. As revealed in figure 4(a), the input image included a purple region at the bottom that was entirely outside the game layout, which could have negatively affected the model performance. Therefore, in the version-3 model, the image processing method was updated to include only the labyrinth area while removing the extra space, as in figure 4(b). Also, hyperparameters were modified accordingly to allow for more exploration.

As a result, the third model yielded a better result, as shown in figure 5, with a moving average reward of around 650. The loss was still large; however, the model could learn a final reward of 1750 near episode 1000, compared to the final reward of only about 1000 in previous two versions of the model. With more computational resource and training episodes, the moving average reward was believed to reach even higher.



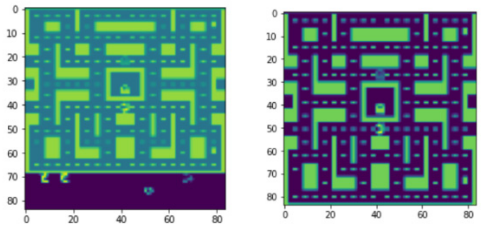
(a) (b)

Fig 2. Graph of simple DQN in Ms. Pacman for (a) episodic reward, and (b) episodic loss.



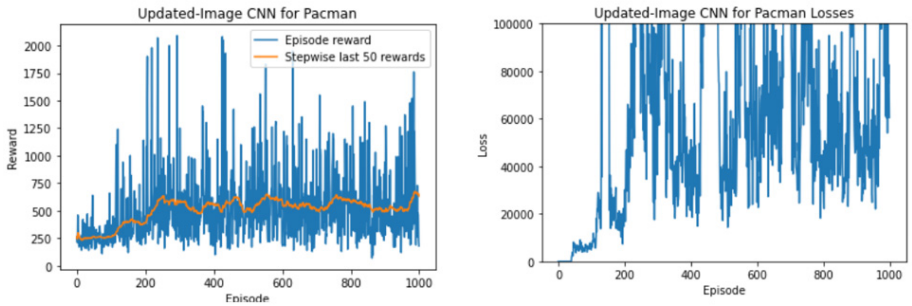
(a) (b)

Fig 3. Graph of complex DQN in Ms. Pacman for (a) episodic reward, and (b) episodic loss.



(a) (b)

Fig 4. (a) Previous input image with a large chunk of purple region below the layout, and (b) updated input image with the extra region removed.



(a) (b)

Fig 5. Graph of DQN with the updated image processing in Ms. Pacman for (a) episodic reward, and (b) episodic loss.

4. Conclusion

In a nutshell, the DQN model can learn the simple Atari game Pong well, reaching a decent performance after 1000 episodes of training with a smaller replay buffer size and reduced epsilon. For complicated games such as Ms. Pacman, however, the model can still obtain reasonably good reward, even though the model performance is not comparable to that in Pong due to diverged loss. Attempts are taken to address the issue by making the model deeper and removing the redundant region surrounding the input image, and despite no notable improvement in loss, some increase in moving average reward is observed. Given the limited time and computational resource, a thorough search of optimal hyperparameters is not conducted. Still, it is believed that the model can be further improved. Future work can examine and experiment on some other models, such as Double Q Network and Actor-Critic Method.

References

[1] Tesauro, G. TD-Gammon: A Self-Teaching Backgammon Program. Applications of Neural Networks, 1995, pp. 267–285.

[2] Mnih, V, et al. Playing Atari with Deep Reinforcement Learning. ArXiv, 2013.

[3] Long-Ji, L. Reinforcement learning for robots using neural networks. Carnegie Mellon University, 1992.

[4] Bellemare, M, et al. The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research, 2013, 47, pp. 253-279.