2024

Deep Q-learning Report

MOSTAFA ELGHAZAWY

Table of Contents

Introduction *1*

Learning Algorithm *1*

Algorithm Description **1**

Hyperparameters **3**

Neural Network Architecture **3**

Results *4*

Ideas for Future Work *4*

References *5*

Introduction

This is a report that describes an implementation for a deep reinforcement agent that solves the banana collector environment. In the banana collector environment, the agent is supposed to collect yellow bananas and avoid blue bananas and the more yellow bananas it collects and the fewer blue bananas it collects, the higher the rewards. If the average score over 100 consecutive episodes is higher than 13, the environment is considered solved. The report will first discuss the algorithm that was used along with the chosen hyperparameters and the neural network model architecture. Then, it will discuss the results of the algorithm.

Learning Algorithm

Algorithm Description

The learning algorithm that was used is deep Q-network or also called Deep Q-learning. Deep Q-learning is very similar to normal Q-learning in reinforcement learning, but instead of using a table, it tries to approximate the table using neural networks as shown in Figure 1.

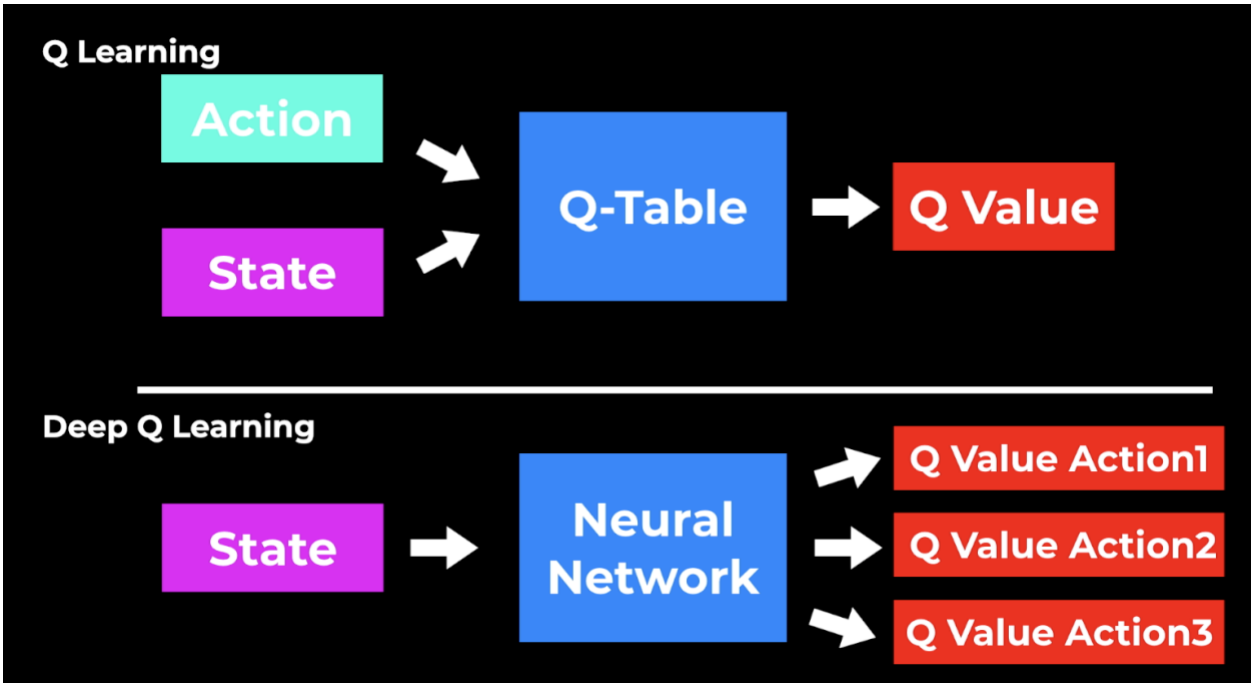

Figure 1: Differences between Q-Learning and Deep Q-Learning [1]

The reason why it approximates the Q-Table with a neural network is that the environment has a continuous state space and is not discrete so it would be impossible to just write in a table like before.

To also make the model more robust, the techniques of "experience replay" and "Fixed Q-targets" were used. In experience replay instead of the model collecting the experience and learning from it and then discarding it, it instead saves the experiences in a buffer and then chooses an experience to learn at random, this makes the model learn from rare experiences more often which makes the model more robust. Also, sequential data can be highly correlated, but experience replay helps break that correlation by sampling at random and not just learning from experiences sequentially.

The model also uses Fixed Q-targets. Before speaking about Fixed Q-targets, let's first speak about the issue it is trying to fix. Generally, in deep learning, there is a predicted output by the neural network and there is the target output. In normal deep-learning applications, the target output is fixed, making it easier for the model to converge. However, in deep Q-Learning, there is the issue that both the predicted output and the target output are changing, because it uses the TD target as the target which also uses the model parameters to be predicted (this is like updating a guess with a guess). So, to fix this issue, Fixed Q-targets are used where the model uses old model parameters and fixes them for some time so that the model can learn from a fixed target like a normal deep learning application, then after that period passes, the target parameters get updated with the new parameters, making the learning process less oscillating and diverge better.

Hyperparameters

- Epsilon starting value = 1.0, Epsilon ending or minimum value = 0.01, Epsilon decay = 0.995

The reason why Epsilon's ending or minimum value is not zero is to ensure that the model will always have a chance to explore the environment. Also, Epsilon decay was set to a high value close to one to ensure it will keep learning for a long period of time.

- Experience replay buffer size = 1e5

It is set to a high value to ensure that the correlation in sequential input is broken as hard as possible and also to ensure the algorithm will keep learning from rare past experience.

- Batch size = 64

This value depends mainly on the GPU used, and the bigger the GPU the better to have a bigger number to ensure that the GPU resources are used fully for faster network learning

- Learning rate = 5e-4

The learning rate is set for a small value to ensure that the model will converge to the optimal value and not just keep oscillating. But at the same time not extremely low so that the learning doesn't become extremely low

- Discount factor gamma = 0.99

If it was set to 1, the model would consider the instant reward as important as future rewards, and this is not the ideal case most of the time because of two reasons. First, the model is less certain about future rewards. Second, it would be better to get the same expected return faster. So, the gamma is set for a value slightly lower than 1 to ensure that instant rewards are given more value than future rewards in calculating expected return.

Neural Network Architecture

The network has 4 fully connected layers. The first three have Relu activation function as output for ensuring non-linearity approximation. The last layer doesn't have any activation because the output of the neural network should be the Q-value for each action, and this can be positive or negative and not values between 0 and 1 so both Relu and Softmax wouldn't work.

The first fully connected layer is named fc1 and takes the state as input, so it has the same size input as the state size and gives an output equal to 128. Then, it goes to the second and has the same 128 size as input and output. Then, the final fourth layer takes in the input of the third layer (size 128) and gives an output of the action size because it gives a state value for each action.

Results

The learning was done in only 429 episodes and has gotten the intended results of an average score higher than 13 per consecutive 100 episodes as clear in Figure 2.

```
Episode 100     Average Score: 0.95
Episode 200     Average Score: 4.22
Episode 300     Average Score: 7.33
Episode 400     Average Score: 10.20
Episode 500     Average Score: 12.41
Episode 529     Average Score: 13.05
Environment solved in 429 episodes!     Average Score: 13.05
```

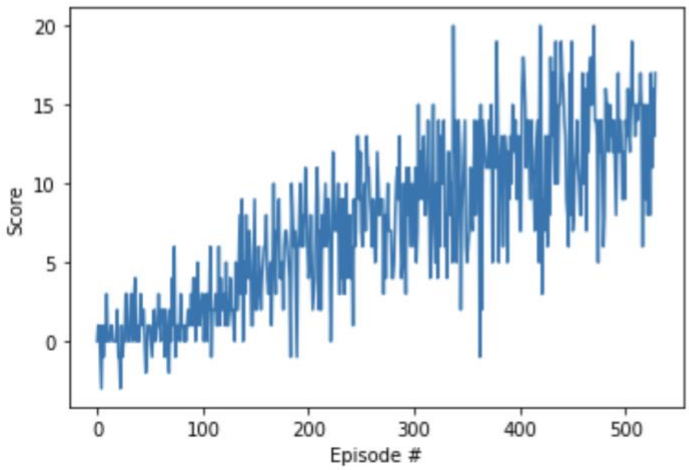

Figure 2: Training Results

Also, when the model is run on the trained parameters, it collects the yellow bananas and avoids the blue ones which shows it was trained properly.

Ideas for Future Work

I highly recommend that the next improvement for the model would be to include the "Rainbow DQN" as the algorithm since it usually gives better results as in Figure 3

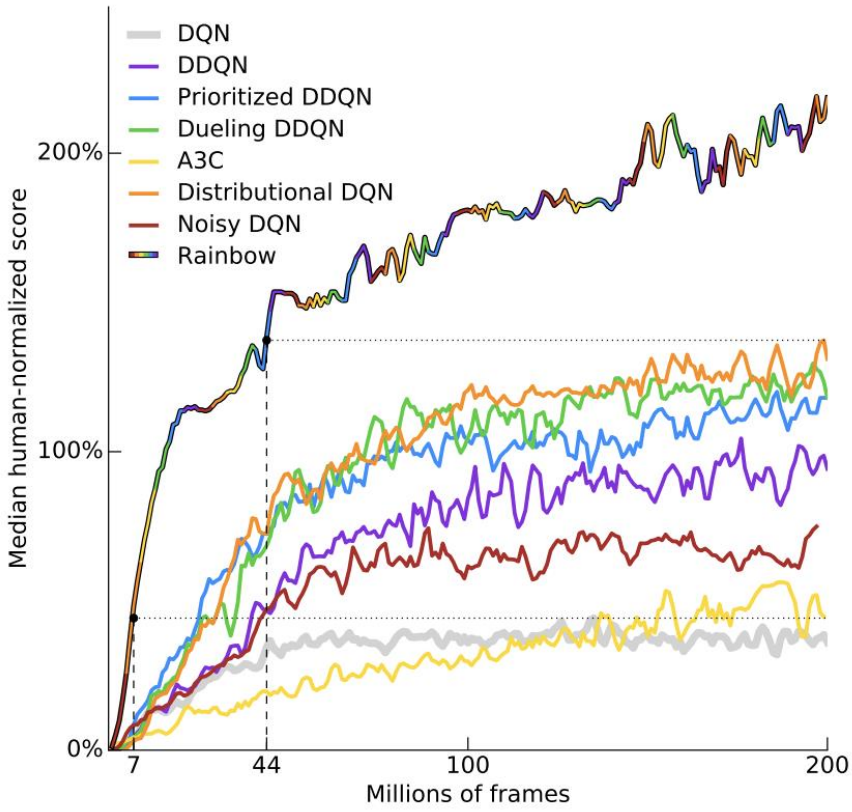

Figure 3: Rainbow DQN results comparison [2]

Also, try to have the network learn faster in fewer episodes by changing the parameters of the network or hyperparameters.

You can also try to have the model learn from pixels as if it is human learning by just watching the monitor which would be more challenging.

References

[1] Loeber P. Reinforcement learning with (Deep) Q-Learning explained [Internet]. Tutorials, AI Research. 2022. Available from: https://www.assemblyai.com/blog/reinforcement-learning-with-deep-q-learning-explained/

[2] Hessel M, Modayil J, Hado VH, Schaul T, Ostrovski G, Dabney W, et al. Rainbow: combining improvements in deep reinforcement learning [Internet]. arXiv.org. 2017. Available from: https://arxiv.org/abs/1710.02298v1