DQN for Navigation

Abdelkarim Eljandoubi

September 23, 2022

1 Introduction

In this project, I trained an agent to solve the banana environment for two categories of state space. The goal is to gather as many yellow bananas (of reward $+1$) as possible while avoiding blue bananas (of reward -1). To do so, the agent has to choose from four actions :

0 move forward.

1. move backward.
2. turn left.
3. turn right.

In order to maximise its accumulative reward given a state. The banana environment has the two kinds of state space :

- vector which has 37 dimensions and contains the agent's velocity, along with ray-based perception of objects around the agent's forward direction.
- RGB image of size 84×84 , corresponding to the agent's first-person view of the environment.

The environment is solved if the agent can achieve a score of $+13$ over 100 consecutive episodes.

2 Implementations

I have started from the code provided in Udacity as a solution for the coding exercise which implements the basic form of DQN [1] for the agent. On the one hand, I added a PyTorch implementation of double DQN to the agent class. On the other hand, [3], I adapted the idea of prioritized experience replay from [2] to the class ReplayBuffer.

2.1 Vector state

I used here the same architecture from the coding exercise : two fully connected hidden layers both with 64 units and followed by a rectified linear unit. The final layer is fully connected with the action state size (4) units.

2.2 Image state

Here, I tested some architectures:

1. The Convolution neural network from DQN paper [1]
2. The Convolution neural network from dueling DQN paper [4]
3. Modified dueling DQN. I made a minor change between the final hidden layer and output layer : an additional fully connected layer with 512 units followed by a ReLU activation function.

3 Results

Throughout experiences, I set

- batch size to 64
- the target update frequency to 4
- learning rate to 5×10^{-5}
- $\gamma = 0.99$
- $\tau = 10^{-3}$
- $\epsilon = 10^{-9}$

Also, I trained the agent by the Adam optimizer with double DQN because of its superior experimentally performance over vanilla.

3.1 Vector state

The agent succeeded to solve the environment with only 500 episodes (see fig. 1), replay buffer of size 10^5 and no prioritized replay ($\alpha = \beta = 0$).

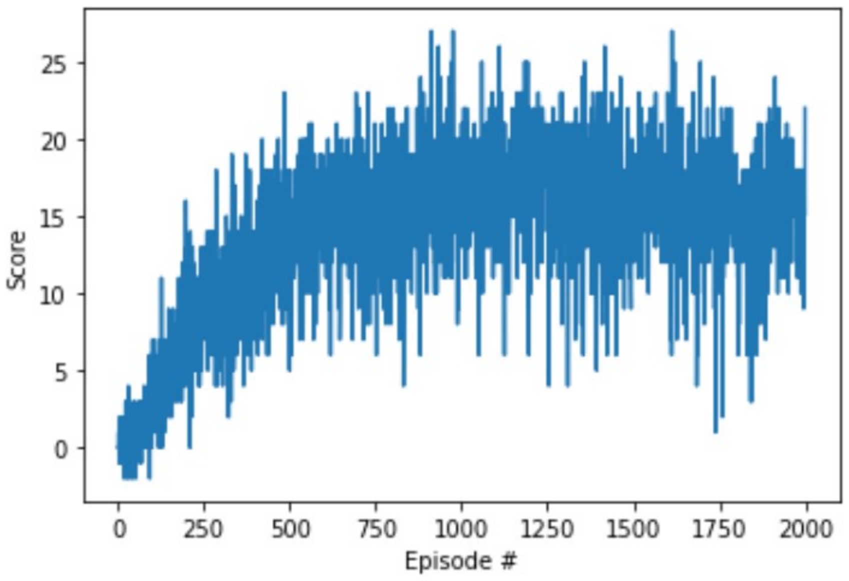

Figure 1: The score plot for the vector state.

3.2 Image state

First, I decreased the replay buffer size to 3×10^4 due to the size of the new state (3D tensor). Then, I tried three types of image transformers ϕ:

ϕ_1 from RGB to grey scale

ϕ_2 from RGB to YB where Y (yellow) is the mean between R (red) and G green.

ϕ_3 identity

Note that ϕ_i has i channel output.

I consider a state as $s_i = (\phi(x_j))_{i-n\_frames \leq j \leq i}$ where x_j the frame (RGB image) at timestamp j and n_frames number of frames per state experimentally equals 4. s_i is a 3D tensor having a size of

$n \times 84 \times 84$ for $n = n\_frames \times$ number of channels of ϕ. As expected, the outcome of ϕ_3 surpasses ϕ_i for $i = 1, 2$. In addition, the modified dueling DQN model beats DQN and dueling DQN in this game. Finally, the chosen prioritised replay parameters are $\alpha = 1.0$ and $_beta = 1.0$. In training, the highest score, I managed to get, is $12, 63$ after 1200 episodes and then the score deceased (see fig. 2). However, when I tested the policy, it did not work.

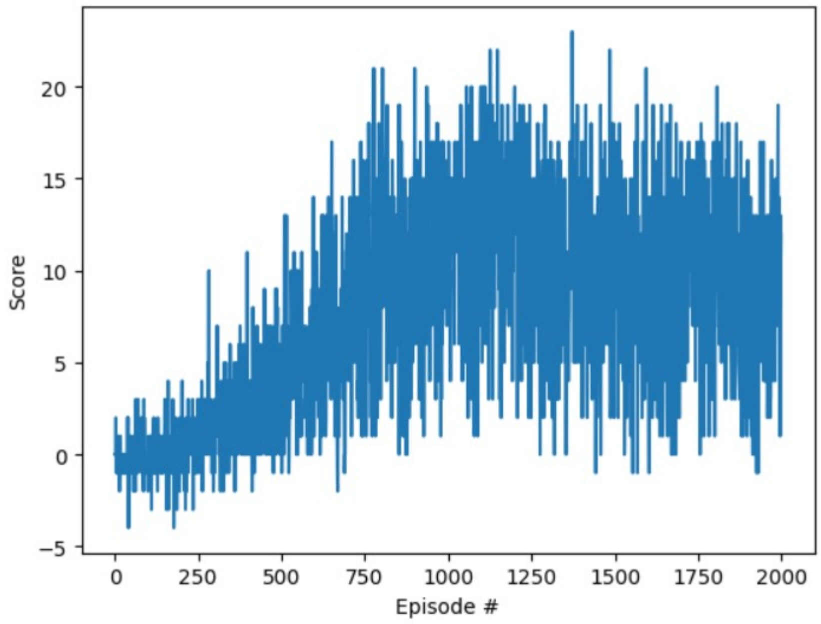

Figure 2: The score plot for the matrix state.

4 Future work

In addition to the current work, we can do the following to improve performance of Navigation Pixels by:

- Compose :
  1. a classifier network as ResNet that we trounce at some level
  2. a hidden linear layer with an output size of $state\_size$
  3. the network of vector state navigation (see section 2.1)
- Use transfer learning on both parts 1 and 3 of the previous architecture.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[2] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015. cite arxiv:1511.05952Comment: Published at ICLR 2016.

[3] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[4] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.