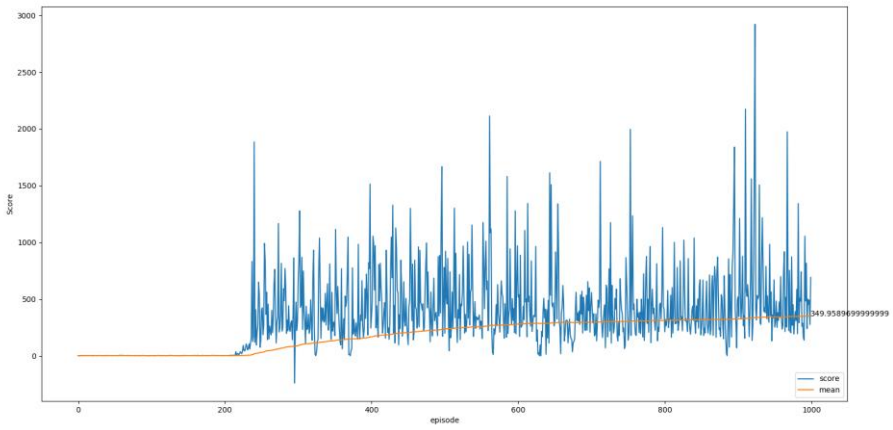
LAB8 : DQN + DDPG

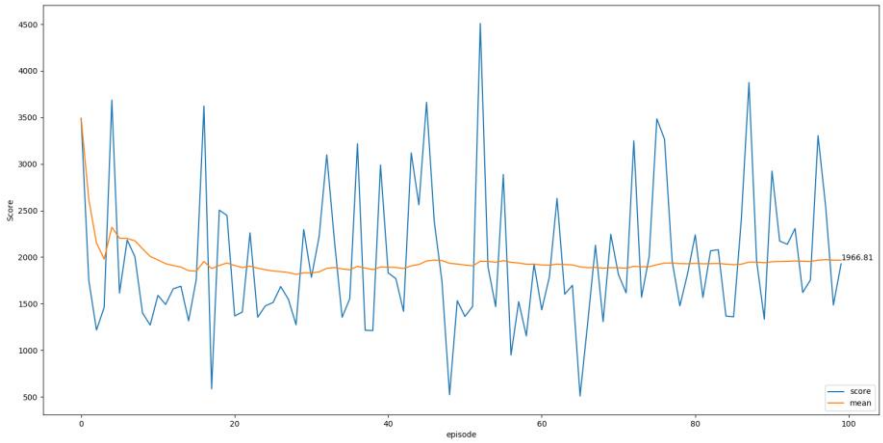
0516069 翁英傑

1. Result

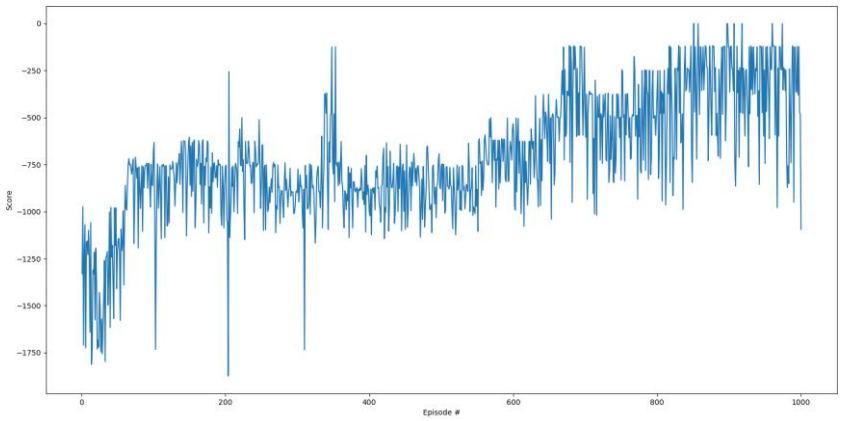
◦ Result for training 1000 episodes with DQN learning.



◦ Test for 100 episodes with epsilon random action (it won’t end without it)



◦ DDPG with highest score around 0 with 1000 episode.



2. Network structure & each loss function

- DQN
  - 2 network with the same structure with input of 4 dimension of information of current state, and output of 2 dimension for two action that can be made. Hidden layer can be any number, here we set 32.
  - Loss function : Adam.

- DDPG
  - Critic
    - 2 network with the same structure. Input with 3 dimension of information of current state and concatenate an addition dimension for action from actor. Output with 1 dimension that gives the value of this action in this state.
  - Actor
    - 2 network with the same structure. Input with 3 dimension of information of current state. Output with 1 dimension that gives the best action which gets the highest score from critic.
  - Loss function : Adam.

3. Training process of deep Q-learning

The goal of DQN is to train a network that can output score for every action with a state as input. First, we took random steps to fill the replay buffer. Then we can start the training with updating the network to gradient to calculate the correct value of taking action in certain state and also update the replay buffer to make sure it is going the correct direction of gradient.

4. Epsilon-greedy action select method

Because we might always choose the same action if it was found out to gain positive rewards in early tries and will ignore trying other steps which might have much more reward. To solve this problem, we make sure the policy choose random actions in a certain proportion so that new possibilities can be tested.

5. Critic updating

We update the critic with almost the same way as Q network in DQN, which is covered by the previous section. Different from previous is that the action is given by the actor network rather than the maximum value of action. Also the update of the target network, which will be discussed in section 7, is soft update

rather than DQN's hard update. It is a weighted sum of both network.

6. Actor updating

Actor network is a network that stimulates a policy. Updating the actor network is to maximize the output of critic network, which can be seen as to maximize a function J(θ). This is the same as doing policy gradient. Repeating previous step and this step until we get a decent actor.

7. Calculate the gradients

Problem may cause if me are trying to find the lowest point of a continuous changing equation, which may not converge. Here we create a fixed network so that the gradient will not be changed, but also this network has to synchronize with the network that keeps updating in a couple of episodes. That is why we have 2 networks for DQN and 2+2 networks for DDPG. There are mathematical proves that it will converge in this way.

8. Code

The codes are both strictly followed by the algorithms shown in the spec. It would be waste of time if I explain whole of my code line by line in this report. Please forgive my absence of this section.

9. Improvement

The reward is slightly modified in the DQN code, which takes the angle into count so that the system know vertical is better thus learns a more robust policy that doesn't ends.