Applied Deep Learning Homework 3

學號：B04901070
系級：電機四
姓名：蘇峯廣

Q1: Basic Performance (6%)

1. Describe your Policy Gradient & DQN model (1% + 1%)

(1) Policy Gradient

$$Policy\ gradient: E_\pi[\nabla_\theta(\underbrace{log\pi(s,a,\theta)}_{\text{Policy function}})\underbrace{R(\tau)}_{\text{Score function}}]$$

$$Update\ rule: \underbrace{\Delta\theta}_{\text{Change in parameters}} = \underbrace{\alpha}_{\text{Learning rate}} * \nabla_\theta(log\pi(s,a,\theta))R(\tau)$$

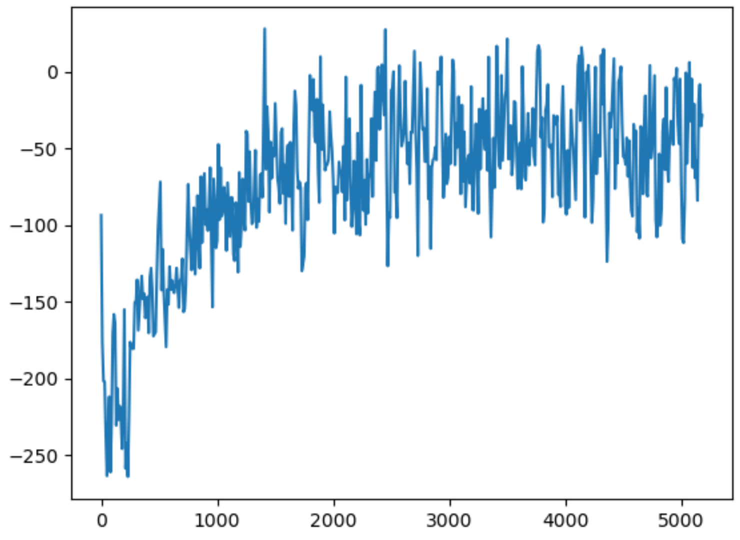
I first used PolicyNet to get the probability from the current state, and used torch.distributions.Categorical to sample an action from the probability. Then, by discounting the saved loss and using the loss function shown above, the model can update the parameters of the PolicyNet.

(2) DQN model

$$\underbrace{\Delta w}_{\text{Change in weights}} = \underbrace{\alpha}_{\text{learning rate}}[\underbrace{(\underbrace{R+\gamma\, max_a\, \hat{Q}(s',a,w))}_{\text{Maximum possible Qvalue for the next\_state (= Q\_target)}} - \underbrace{\hat{Q}(s,a,w)}_{\text{Current predicted Q-val}}}_{\text{TD Error}}]\ \underbrace{\nabla_w\hat{Q}(s,a,w)}_{\text{Gradient of our current predicted Q-value}}$$

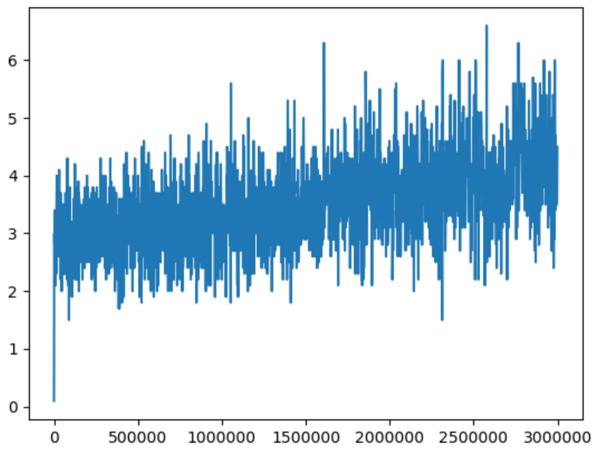
There are two PolicyNet used here. I first used the online_net to get the probability from the responding action. If exploration occurred, I randomly sampled an action. Otherwise, I sampled an action like what PG model had done. The exploration schedule here is 1.0 + (1.0 – 0.01) * math.exp(-1. * current_step / 3000000). So the model would tend to explore at first and gradually used the model more to predict. Then, by sampling the buffer, discounting the saved loss and using the loss function shown above, the model can update the parameters of the PolicyNet.

2. Plot the learning curve to show the performance of your Policy Gradient on LunarLander (2%)



We can find that the average reward is larger and larger as the number of time steps grow. Therefore, it is clear that the PolicyNet model can learn from the environments and therefore does a right decision..

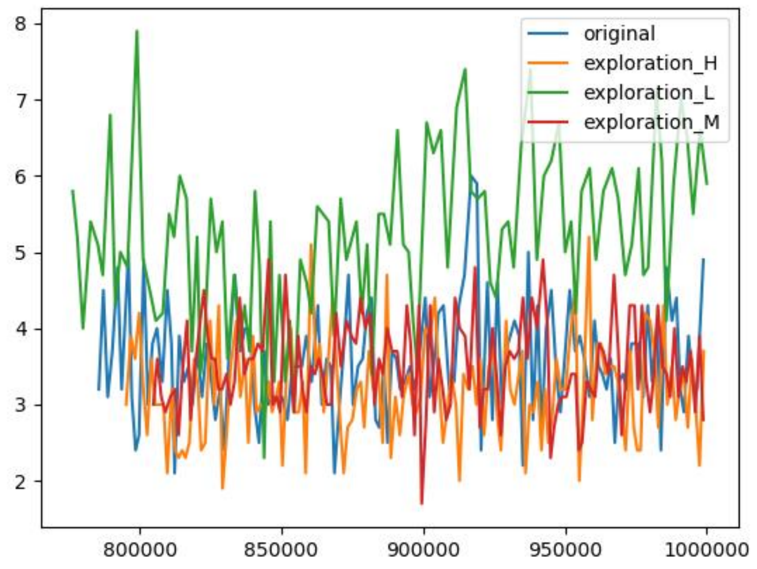
3. Plot the learning curve to show the performance of your DQN on Assualt (2%)



Not clear as it may be, the average reward still grows at a slow pace. I think that the reason is the clipping rewards. But at the testing step, the it is quite obvious that the PolicyNet model can learn from the environments and therefore does a right decision.

Q2: Experimenting with DQN hyperparameters (2%)

1. Plot all four learning curves in the same figure (1%)



- Original：Exploration epsilon = 1.0 – 0.01 ; gamma = 0.99
- Exploration_H：Exploration epsilon = 1.0 – 0.7 ; gamma = 0.99
- Exploration_L：Exploration epsilon = 0.3 – 0.01 ; gamma = 0.99
- Exploration_M：Exploration epsilon = 0.3 – 0.7 ; gamma = 0.99

We can find that the average reward is the highest on Exploration_L model. Moreover, it is apparent that the original model is the most stable. At last, we can find that the Exploration_M model is the least stable.

2. Explain why you choose this hyperparameter and how it affect the results (0.5% + 0.5%)

I choose these settings because I want to analyze that the relations between exploration schedule and the result. What would happen if the model considers more/less about exploration? And the reason why choosing the Exploration_M model setting is to find that if the epsilon is not change too much in the process, what would happen.

- Original：Averaging reward in 100 episodes=124.74
- Exploration_H：Averaging reward in 100 episodes=95.76
- Exploration_L：Averaging reward in 100 episodes=146.54
- Exploration_M：Averaging reward in 100 episodes=76.65

As for the original model, the result is not bad by the way, but there are still better models for this task. As for the Exploration_H model, it doesn't work well because it doesn't take too much the PolicyNet model prediction into account. As for the Exploration_L model, it works surprisingly well here, and I think the reason here is that exploration is rather less necessary for this task because Assault is quite a straight-forward task. And as for the Exploration_M model, little change in epsilon (nearly 0.5) makes fluctuations at the training step, which directly influences the average reward at testing step.

Q3: Improvements to PG & DQN / Other RL methods (2%+2%)

Choose two improvements to PG & DQN or other RL methods

- Normal dqn on Assault：100.06
- Double-dqn on Assault：133.03
- Dueling-dqn on Assault：132.30

[PS] All models use the same setting in this experiment to ensure that the model really improves.

1. Describe why they can improve the performance (1%)

(A) Double-dqn

$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s'\sim D}\left[\left(r + \gamma\hat{Q}(s', \arg\max_{a'} Q(s', a', w), w^-) - Q(s, a, w)\right)^2\right]$$

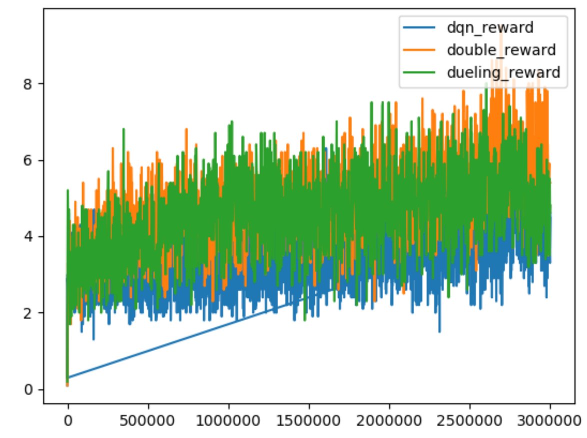
The advantage of double-dqn is that it uses two networks to decouple the action selection from the target Q value generation when we compute the Q target, which helps us reduce the overestimation of q values and, as a consequence, helps us train faster and have more stable learning to get better results.

(B) Dueling-dqn

$$Q(s, a) = V(s) + A(s, a)$$

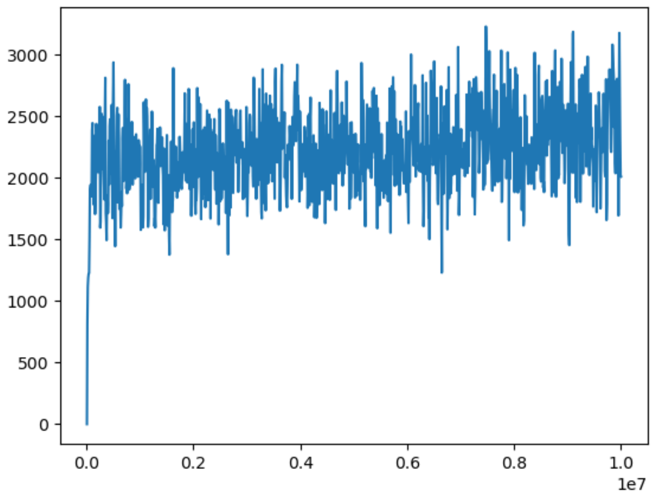
The advantage of dueling-dqn is that it decomposes Q-function into advantage and value function, which avoids the network being overoptimistic. Otherwise it will cause actual DQN and to get as optimistic as it explodes.

2. Plot the graph to compare results with and w/o improvement (1%)

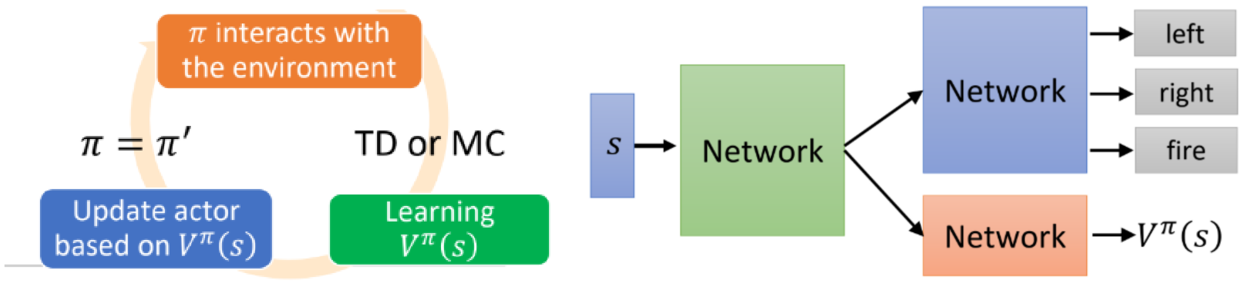


Q4: Bonus.

1. Plot the learning curve to show the performance.



2. Describe the RL algorithm you used.



I use A2C(policy gradient along with dqn) in my implementation. TD and entropy regularization are used here, but I don’t use RNN in this A2C model.