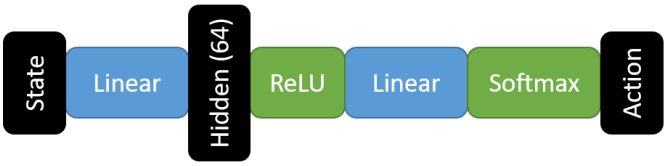
Applied Deep Learning HW3-Report

B04901020 電機四 解正平

Q1：Baseline Performance (6%)

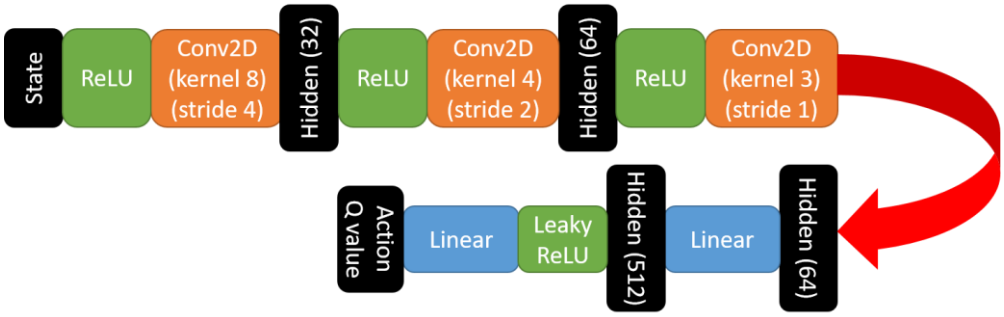
1. Describe your Policy Gradient & DQN model (2%)

   Policy Gradient model



   The model for policy gradient is to decide the action with state. I only use simple linear layer to build the network, as shown in the above. Besides the architecture, I calculate the discounted reward for each time stamp and give zero accumulated rewards if encounter zero reward.

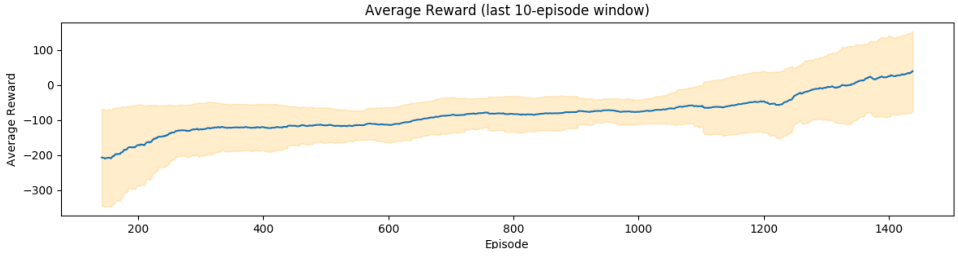
   DQN model



   The model for DQN is to predict the future rewards. I use convolution network to extract features from the screen image and use linear layer to acquire the Q value for each actions. Moreover, I use epsilon greedy as the exploration algorithm, which the epsilon value is linear decayed and will not be zero until exceeds a threshold step.

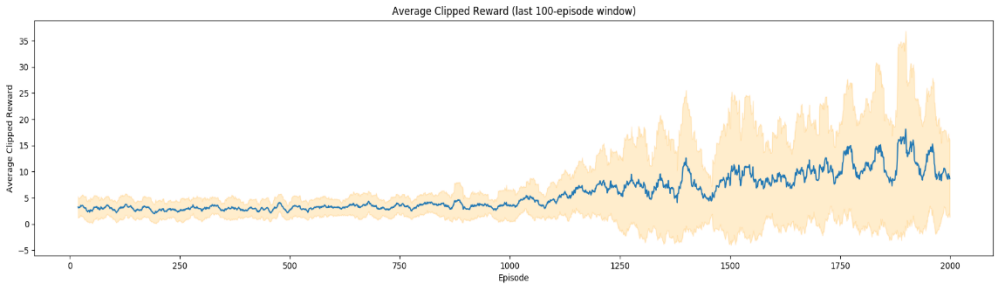
2. Plot the learning curve to show the performance of policy gradient (2%)

   We can see the average reward (last-10) for each episodes. The performance is more unstable at first but goes higher and stable in the middle of training. In nearly 1500 episodes, we can beat the baseline.

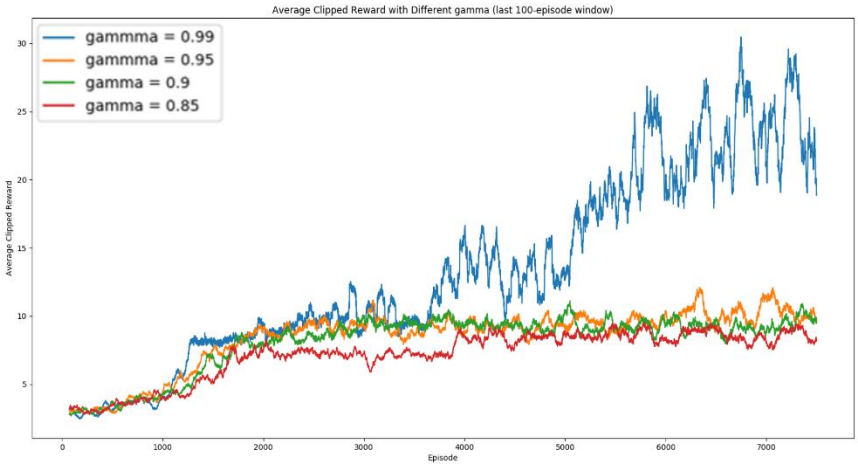


3. Plot the learning curve to show the performance of DQN (2%)

DQN is at exploration phase in the beginning; however, it turns to learning phase in approximately 1000 episodes. We can find the average reward (last 100) goes higher and oscillated while the rewards are clipped between (-1, 1) to keep model more stable. In nearly 2000 episodes, we beat the baseline.



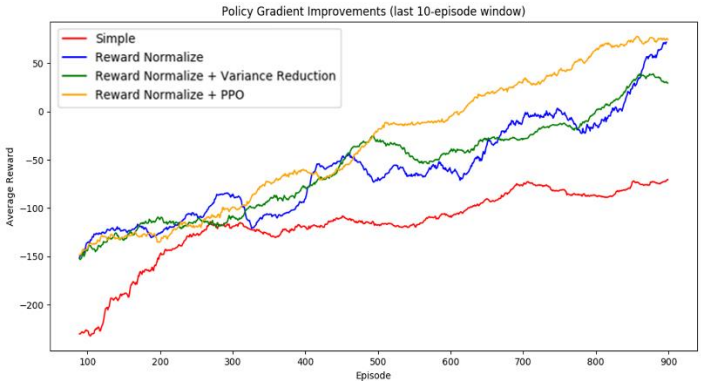
Q2：Experiment with DQN hyper-parameters (2%)



The environment I use is Assault Game. I choose gamma as the experiment hyper-parameters due to its ability to consider future rewards and affect the performance obviously. The results show that the higher of gamma, the more emphasized for future. If we give small gamma value, we cannot see future rewards and lead to a bad performance. Therefore, we need to choose gamma = 0.99 as the paper described.

Q3：Improvements to Policy Gradient, DQN / Other RL method (4%)

1. Policy Gradient Improvements (2%)



I implement several improvements for policy gradient on LunarLander Game, including reward normalization, variance reduction, and proximal policy optimization (PPO).

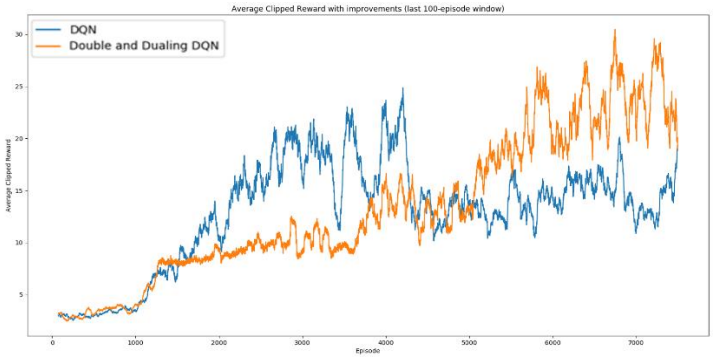
Due to rewards may all positive, we can subtract a baseline (normalization) to let rewards have negative value, which show a great improvement. With baseline, the probability of the not sampled actions will not decrease sharply. Besides above, we can set specific baseline to reduce the variance as followed. However, this method has little influence on the normalized rewards.

$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]}$$

$$J_{ppo2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \min\left(\frac{P_\theta(a_t|s_t)}{P_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t,a_t), clip\left(\frac{P_\theta(a_t|s_t)}{P_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t,a_t)\right)$$

Moreover, the best method in this years is PPO, which had implement off-policy algorithm with important sampling. PPO set KL divergence constraints for θ cannot very different from θ'. The objective function is showed above and we can see it obtains the best performance.

2. DQN Improvements (2%)



I implement double and dueling DQN for the improvements on Assault Game. It is a simple method only need to use another (online) network to

acquire target actions for double DQN while modify the network structure for dueling DQN.

Double DQN

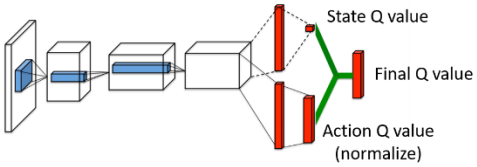
$$Q(s,a) = r(s,a) + \gamma Q(s', argmax_a Q(s',a))$$

TD target

DQN Network choose action for next state

Target network calculates the Q value of taking that action at that state

Dueling DQN

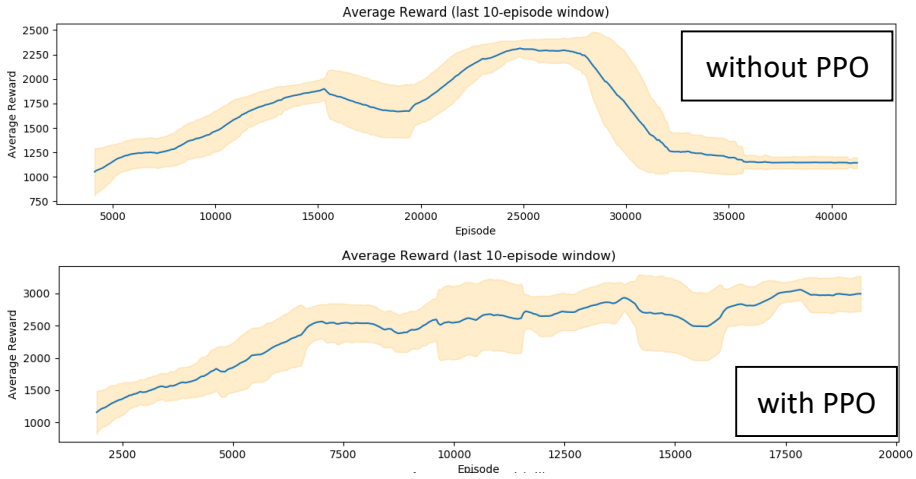


Double DQN is used to solved over-estimated problems. With two networks (online and target), they can compensate for the other to avoid over-estimated q value. Dueling DQN is used to acquire the state q value among each actions and set a normalized constraint for different action q value. With this method, we can update the action even if we don't sample on it, which is more efficient.

In my experiment, we set exploration phase with 1000 episodes. We can find DQN has higher performance at first but double and dueling has the best results afterwards. The basic DQN converge fast than improved DQN but cannot enhance the received rewards. With double and dueling DQN, we can still get higher scores when using small exploration steps.

Bonus (4%) :

1. Plot



From the above image, we can find that the performance with PPO is higher than without PPO. The method with PPO suggests a better result with a fast and effective improvement. However, the performance will go decayed with more training steps. Besides, the variance of rewards (yellow part) shows the unstable on reinforcement learning. It is difficult to train a good agent.

2. Algorithm (Ref-code: https://github.com/higgsfield/RL-Adventure-2/blob/master/3.ppo.ipynb)

   I implement simple A2C algorithm with proximal policy optimization (PPO) and Generalized Advantage Estimation (GAE) on multi-processing environment with value loss, action loss, and entropy loss. However, I didn't use the RNN network in my methods due to the bad performance in my experiment. The method can consider the KL-divergence constraints and train more iteration on one steps.