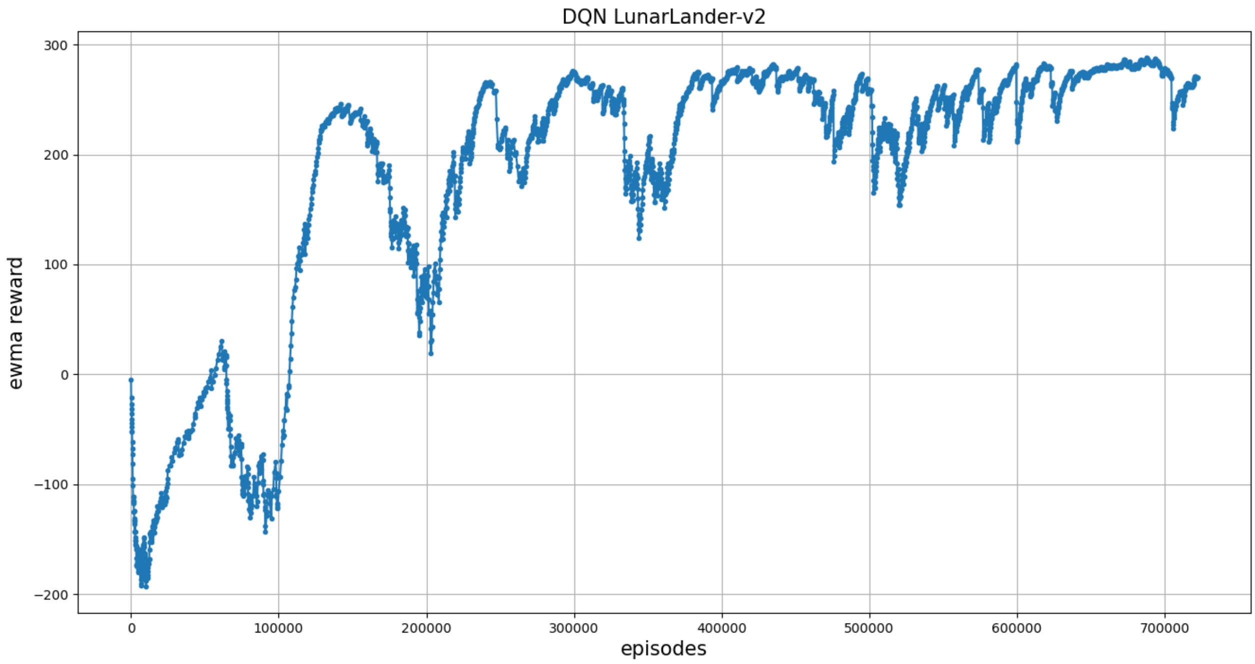
Deep Learning and Practice

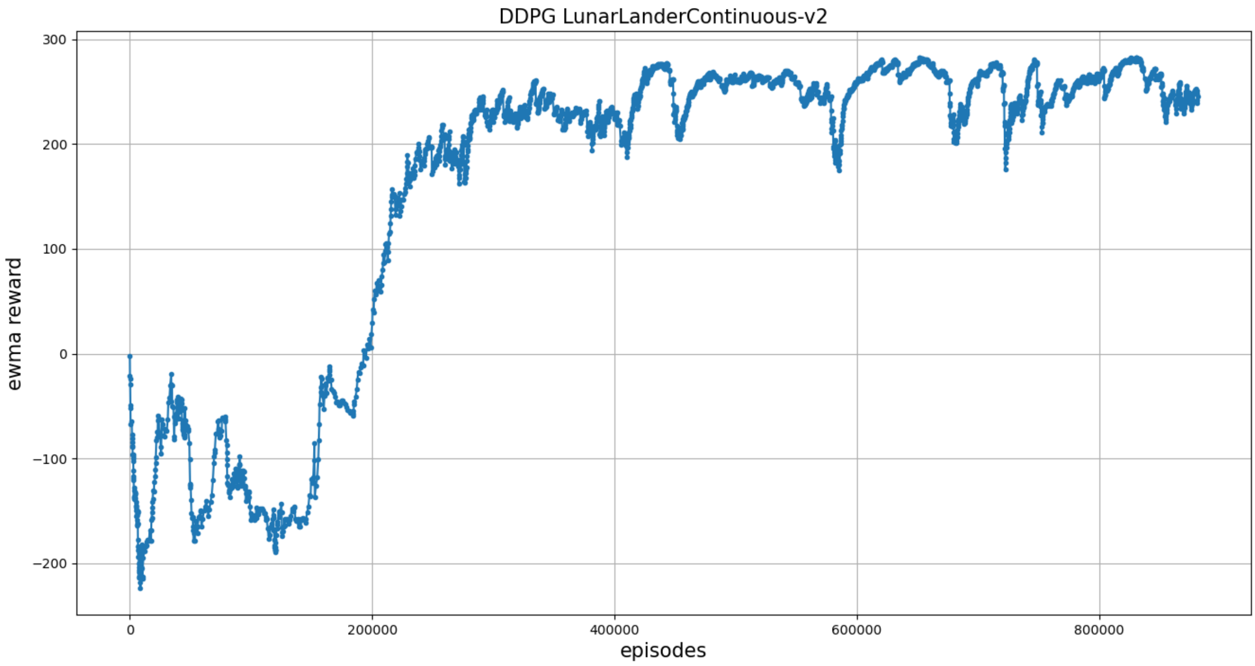
Lab 6

309505018 郭俊廷

1. A plot shows episode rewards of at least 800 training episodes in LunarLander-v2 (5%)



2. A plot shows episode rewards of at least 800 training episodes in LunarLanderContinuous-v2 (5%)



3. Describe your major implementation of both algorithms in detail.(20%)

Deep Q-Learning (DQN):

Deep Q-Learning是一個model-free, off-policy的演算法, 跟一般Q-Learning最大的差別就是用neural network取代Q-table, 所以init 這裡先建一個network預測Q value, 以LunarLander-v2這個環境來說有四個action (No-op, fire left engine, fire right engine, fire main engine)所以最後 fully connect有四個output, 即action_dim=4, 再通過一層ReLU取正的值

```
class Net(nn.Module):
    def __init__(self, state_dim=8, action_dim=4, hidden_dim=(400,300)):
        super().__init__()
        ## TODO ##
        self.fc1=nn.Linear(state_dim,hidden_dim[0])
        self.fc2=nn.Linear(hidden_dim[0],hidden_dim[1])
        self.fc3=nn.Linear(hidden_dim[1],action_dim)
        self.relu=nn.ReLU()

    def forward(self, x):
        ## TODO ##
        out=self.relu(self.fc1(x))
        out=self.relu(self.fc2(out))
        out=self.fc3(out)
        return out
```

在select action時採用 $\varepsilon - greedy$ 法, 先random產生一個數字, 如果這個數字小於epsilon, 則隨機採取任意一個action, 如果大於epsilon,則選擇best action

```
#DQN
def select_action(self, state, epsilon, action_space):
    '''epsilon-greedy based on behavior network'''
    ## TODO ##
    if random.random() < epsilon:
        return action_space.sample()
    else:
        with torch.no_grad():
            return self._behavior_net(torch.from_numpy(state).view(1,-1).to(self.device))
                                                            .max(dim=1)[1].item()
```

Update behavior network 是從replay memory中隨機sampling遊戲過程中儲存的 Next state, current state, action, reward, ($\phi_t, a_t, r_t, \phi_{t+1}$) 這些資訊來做TD-Learning, 再計算 Q value與Q target的MSE loss

$$Q_{target} = \gamma_j + max\hat{Q}(\phi_{j+1}, a'; \theta^-)$$

Perform a gradient descent step on $(y_j - \hat{Q}(\phi_{j+1}, a'; \theta^-))^2$ with respect to the network parameter θ

```
class DQN:
    def _update_behavior_network(self, gamma):
        # sample a minibatch of transitions
        state, action, reward, next_state, done =
                                self._memory.sample(self.batch_size, self.device)
        ## TODO ##
        q_value = self._behavior_net(state).gather(dim=1,index=action.long())
        with torch.no_grad():
            q_next = self._target_net(next_state).max(dim=1)[0].view(-1,1)
            q_target = reward + gamma*q_next*(1-done)
        criterion = nn.MSELoss()
        loss = criterion(q_value, q_target)

        # bp
        self._optimizer.zero_grad()
        loss.backward()
        nn.utils.clip_grad_norm_(self._behavior_net.parameters(), 5)
        self._optimizer.step()
```

每隔一段時間, 就用behavior network 取代 target network

```
class DQN:
    def _update_target_network(self):
        '''update target network by copying from behavior network'''
        ## TODO ##
        self._target_net.load_state_dict(self._behavior_net.state_dict())
```

Deep Deterministic Policy Gradient (DDPG):

Deep Deterministic Policy Gradient是一個model-free, off-policy的演算法, 他跟DQN的差別有兩個, 第一個是可以解決連續動作的問題, 因為DQN的output是action dim, 是離散的,DDPG的output是連續的, 我是取tanh(x)(如下圖), 第二個是DQN是value based的方法去估計Q(s,a), 但DDPG是actor-critic的方法, 所以有actor network跟critic nework (如下圖)
Actor network的目的是要產生連續動作action,範圍是 -1 ~ +1, 所以取tanh
Critic network的目的是要評估在這個state下,actor所採取的action好不好, 所以他跟QDN很像, 都是根據state action pair去估計value的

```
class ActorNet(nn.Module):
    def __init__(self, state_dim=8, action_dim=2, hidden_dim=(400, 300)):
        super().__init__()
        ## TODO ##
        self.fc1=nn.Linear(state_dim,hidden_dim[0])
        self.fc2=nn.Linear(hidden_dim[0],hidden_dim[1])
        self.fc3=nn.Linear(hidden_dim[1],action_dim)
        self.relu=nn.ReLU()
        self.tanh=nn.Tanh()

    def forward(self, x):
        ## TODO ##
        out=self.relu(self.fc1(x))
        out=self.relu(self.fc2(out))
        out=self.tanh(self.fc3(out))
        return out

class CriticNet(nn.Module):
    def __init__(self, state_dim=8, action_dim=2, hidden_dim=(400, 300)):
        super().__init__()
        h1, h2 = hidden_dim
        self.critic_head = nn.Sequential(
            nn.Linear(state_dim + action_dim, h1),
            nn.ReLU(),
        )
        self.critic = nn.Sequential(
            nn.Linear(h1, h2),
            nn.ReLU(),
            nn.Linear(h2, 1),
        )

    def forward(self, x, action):
        x = self.critic_head(torch.cat([x, action], dim=1))
        return self.critic(x)
```

在select action的時候, action network選擇action並加上noise, 增加隨機性, 因為有時候從既有的state去找是很難找到best action的

```
#DDPG
def select_action(self, state, noise=True):
    '''based on the behavior (actor) network and exploration noise'''
    ## TODO ##
    with torch.no_grad():
        if noise:
            re = self._actor_net(torch.from_numpy(state).view(1,-1).to(self.device))+\
                torch.from_numpy(self._action_noise.sample()).view(1,-1).to(self.device)
        else:
            re = self._actor_net(torch.from_numpy(state).view(1,-1).to(self.device))
        return re.cpu().numpy().squeeze()
```

在training過程中, 要更新behavior的actor network μ, critic network Q, target的actor network μ', critic network Q', 利用target network output的Q_target 與behavior Network output 的 Q value做MSE loss更新Q function

```
## update critic ##
# critic loss
## TODO ##
q_value = self._critic_net(state,action)
with torch.no_grad():
   a_next = self._target_actor_net(next_state)
   q_next = self._target_critic_net(next_state,a_next)
   q_target = reward + gamma*q_next*(1-done)
criterion = nn.MSELoss()
critic_loss = criterion(q_value, q_target)
```

behavior Network 的 critic network output 的 Q(s,a), 我們希望Q(s,a)越大越好, Q(s,a)越大代表在這個state下越適合採取這個action, 因此定義

$$Actor\ Loss = E[-Q(s, \mu(s))]$$

```
## update actor ##
# actor loss
## TODO ##
action = self._actor_net(state)
actor_loss = -self._critic_net(state,action).mean()
```

4. Describe differences between your implementation and algorithms.(10%)

在training的時候, 每一個episode都會有一段warmup時間(10000 steps),在這段時間中不會去update network得參數, 只會隨便探索, 並把探索的 state, next state, reward 存到replay memory裡。

5. Describe your implementation and the gradient of actor updating.(10%)

Actor network 會輸出一個action, critic network會根據這個action輸出一個的Q(s,a), 為了使Q(s,a) value最大, 因此定義actor loss

$$Actor\ Loss = -Q(s, \mu(s))$$

$$\frac{\nabla L}{\nabla \theta_u} = -\frac{\nabla Q(s,a|\theta_u))}{\nabla a} \times \frac{\nabla a}{\nabla u(s|\theta_u))} \times \frac{\nabla u(s|\theta_u))}{\nabla \theta_u} = -\frac{\nabla Q(s,a|\theta_u))}{\nabla u(s|\theta_u))} \times \frac{\nabla u(s|\theta_u))}{\nabla \theta_u}$$

```
## update actor ##
# actor loss
## TODO ##
action = self._actor_net(state)
actor_loss = -self._critic_net(state,action).mean()
```

6. Describe your implementation and the gradient of critic updating.(10%)

利用target network output 的 Q target 與 behavior network output 的 Q(s,a)做mean squre error來更新Q function

$$Critic\ Loss = \frac{1}{N}\sum(Q_{target} - Q(s_t, a_t|\theta_Q))^2$$

```
## update critic ##
# critic loss
## TODO ##
q_value = self._critic_net(state,action)
with torch.no_grad():
   a_next = self._target_actor_net(next_state)
   q_next = self._target_critic_net(next_state,a_next)
   q_target = reward + gamma*q_next*(1-done)
criterion = nn.MSELoss()
critic_loss = criterion(q_value, q_target)
```

7. Explain effects of the discount factor.(5%)

在model free中, 會用未來會得到的reward對現在進行估計, 因此希望越接近現在reward影響越大, 離現在越遠的reward影響比較小, 所以加入discount factor λ

$$G_t = R_{t+1} + \lambda R_{t+2} + \lambda^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \lambda^k R_{t+k+1}$$

8. Explain benefits of epsilon-greedy in comparison to greedy action selection.(5%)

為了確保在explore和exploit之間取得平衡, 因此使用epsilon-greedy, 有ϵ的機率會去explore未知的action, 有$1-\epsilon$的機率會選擇現在最佳的action, 因為如果一直選擇現在最佳的action則有可能找不到比這個更好的action

9. Explain the necessity of the target network.(5%)

Target Network的目的是為了使輸出更穩定, 所以每隔一段時間, 當Q target改變的時候, 就把behavior network 複製到target network, 使輸出更穩定

10. Explain the effect of replay buffer size in case of too large or too small.(5%)

每一個episode都會有一段warmup時間(10000 steps), warmup會儲存到replay buffer, 如果replay buffer size越大, training過程會越穩定, 但會降低training的速度, 如果replay buffer size越小, 容易造成overfitting, 很不穩定, 容易train失敗

11. Bonus - Implement and experiment on Double-DQN (10%)

Double-DQN 跟DQN的差別在於, 在update behavior network是如何決定Q target的, DQN的缺點是估計Q target時會過度估計, Double-DQN在估計Q target不是直接用max Q'(s,a), 而是用Q(s,ai)中最大值的i作為t查找Q'(s,ai)index

DQN:

$$y_j = R_j + \gamma max Q'(\phi(S'_j), A'_j, w')$$

Double-DQN:

$$y_j = R_j + \gamma max Q'(\phi(S'_j), argmax Q(\phi(S'_j), a, w), w')$$

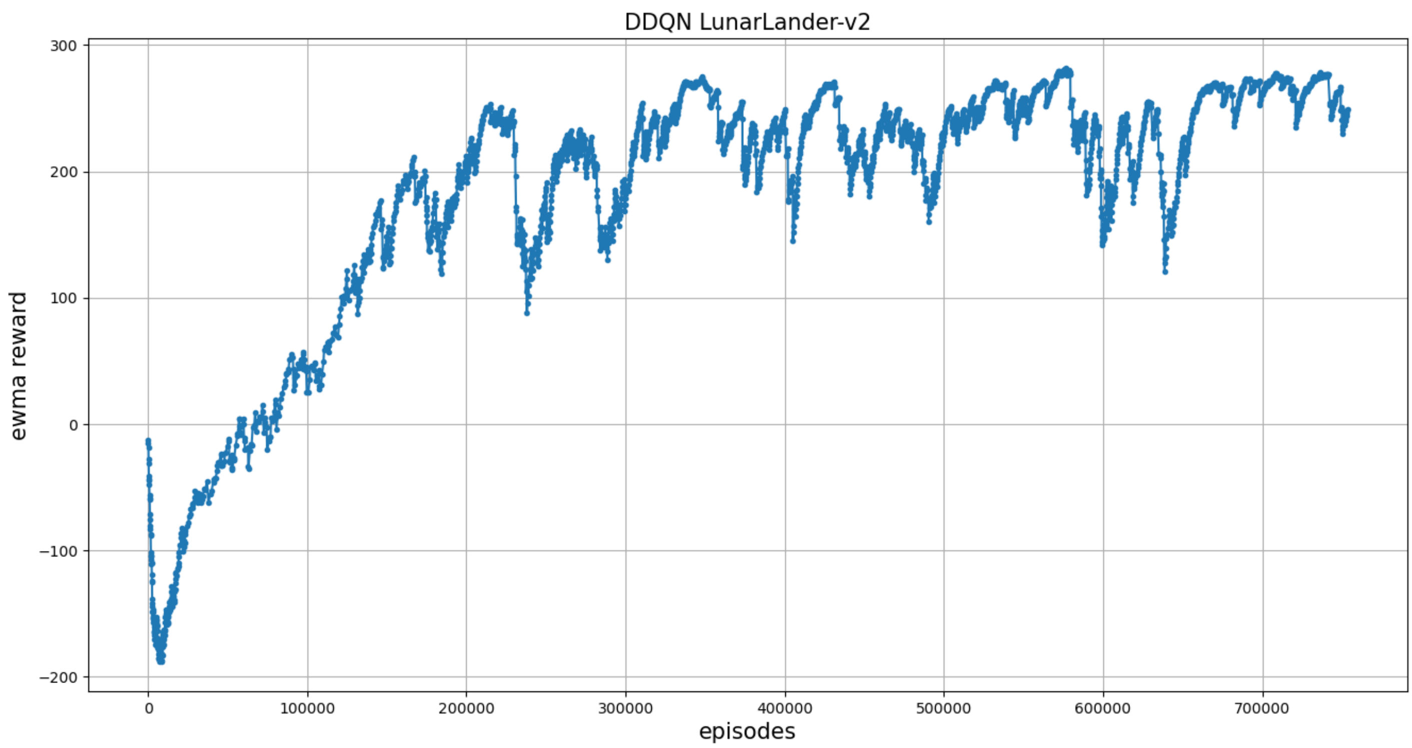
```
class DDQN:
    def _update_behavior_network(self, gamma):
        # sample a minibatch of transitions
        state, action, reward, next_state, done =
                                self._memory.sample(self.batch_size, self.device)
        ## TODO ##
        q_value = self._behavior_net(state).gather(dim=1,index=action.long())
        with torch.no_grad():
            action_index = self._behavior_net(next_state).max(dim=1)[1].view(-1,1)
            q_next = self._target_net(next_state).gather(dim=1,index=action_index;long())
            q_target = reward + gamma*q_next*(1-done)
        criterion = nn.MSELoss()
        loss = criterion(q_value, q_target)

        # bp
        self._optimizer.zero_grad()
        loss.backward()
        nn.utils.clip_grad_norm_(self._behavior_net.parameters(), 5)
        self._optimizer.step()
```

DDQN: Average Reward: 250.845

```
Step: 748660    Episode: 2973    Length: 180    Total reward: 247.85    Ewma reward: 262.27    Epsilon: 0.010
Step: 748867    Episode: 2974    Length: 207    Total reward: 306.52    Ewma reward: 264.48    Epsilon: 0.010
Step: 749059    Episode: 2975    Length: 192    Total reward: 316.95    Ewma reward: 267.11    Epsilon: 0.010
Step: 749232    Episode: 2976    Length: 173    Total reward: 261.13    Ewma reward: 266.81    Epsilon: 0.010
Step: 749330    Episode: 2977    Length:  98    Total reward: -102.56   Ewma reward: 248.34    Epsilon: 0.010
Step: 749495    Episode: 2978    Length: 165    Total reward: 255.28    Ewma reward: 248.69    Epsilon: 0.010
Step: 749707    Episode: 2979    Length: 212    Total reward: 272.98    Ewma reward: 249.90    Epsilon: 0.010
Step: 749897    Episode: 2980    Length: 190    Total reward: 266.13    Ewma reward: 250.71    Epsilon: 0.010
Step: 750052    Episode: 2981    Length: 155    Total reward: 57.25     Ewma reward: 241.04    Epsilon: 0.010
Step: 750163    Episode: 2982    Length: 111    Total reward: 19.37     Ewma reward: 229.96    Epsilon: 0.010
Step: 750363    Episode: 2983    Length: 200    Total reward: 295.22    Ewma reward: 233.22    Epsilon: 0.010
Step: 750600    Episode: 2984    Length: 237    Total reward: 281.03    Ewma reward: 235.61    Epsilon: 0.010
Step: 750775    Episode: 2985    Length: 175    Total reward: 247.76    Ewma reward: 236.22    Epsilon: 0.010
Step: 750972    Episode: 2986    Length: 197    Total reward: 322.36    Ewma reward: 240.52    Epsilon: 0.010
Step: 751291    Episode: 2987    Length: 319    Total reward: 279.22    Ewma reward: 242.46    Epsilon: 0.010
Step: 751450    Episode: 2988    Length: 159    Total reward: 265.70    Ewma reward: 243.62    Epsilon: 0.010
Step: 751598    Episode: 2989    Length: 148    Total reward: 256.47    Ewma reward: 244.26    Epsilon: 0.010
Step: 751761    Episode: 2990    Length: 163    Total reward: 269.11    Ewma reward: 245.51    Epsilon: 0.010
Step: 751940    Episode: 2991    Length: 179    Total reward: 288.63    Ewma reward: 247.66    Epsilon: 0.010
Step: 752056    Episode: 2992    Length: 116    Total reward: 31.64     Ewma reward: 236.86    Epsilon: 0.010
Step: 752293    Episode: 2993    Length: 237    Total reward: 306.00    Ewma reward: 240.32    Epsilon: 0.010
Step: 752507    Episode: 2994    Length: 214    Total reward: 261.92    Ewma reward: 241.40    Epsilon: 0.010
Step: 752680    Episode: 2995    Length: 173    Total reward: 275.34    Ewma reward: 243.09    Epsilon: 0.010
Step: 752861    Episode: 2996    Length: 181    Total reward: 269.91    Ewma reward: 244.44    Epsilon: 0.010
Step: 753077    Episode: 2997    Length: 216    Total reward: 282.25    Ewma reward: 246.33    Epsilon: 0.010
Step: 753356    Episode: 2998    Length: 279    Total reward: 292.02    Ewma reward: 248.61    Epsilon: 0.010
Step: 753503    Episode: 2999    Length: 147    Total reward: 265.00    Ewma reward: 249.43    Epsilon: 0.010
Start Testing
total reward: 245.37
total reward: 270.37
total reward: 278.51
total reward: 262.98
total reward: 29.58
total reward: 271.04
total reward: 282.04
total reward: 280.66
total reward: 292.10
total reward: 295.78
Average Reward 250.84409525944935
(pytorch) jackkuo@lab708-Default-string:~/DLPhw6$
```

Training ewma reward/episode of DDQN:



13. Performance

DQN: Average Reward: 289.75

```
Step: 532439    Episode: 1977   Length: 229    Total reward: 305.00    Ewma reward: 273.35    Epsilon: 0.010
Step: 532641    Episode: 1978   Length: 202    Total reward: 271.09    Ewma reward: 273.24    Epsilon: 0.010
Step: 532816    Episode: 1979   Length: 175    Total reward: 236.59    Ewma reward: 271.41    Epsilon: 0.010
Step: 533000    Episode: 1980   Length: 184    Total reward: 248.19    Ewma reward: 270.25    Epsilon: 0.010
Step: 533209    Episode: 1981   Length: 209    Total reward: 254.25    Ewma reward: 269.45    Epsilon: 0.010
Step: 533409    Episode: 1982   Length: 200    Total reward: 271.11    Ewma reward: 269.53    Epsilon: 0.010
Step: 533604    Episode: 1983   Length: 195    Total reward: 271.61    Ewma reward: 269.63    Epsilon: 0.010
Step: 533785    Episode: 1984   Length: 181    Total reward: 250.72    Ewma reward: 268.69    Epsilon: 0.010
Step: 533984    Episode: 1985   Length: 199    Total reward: 252.11    Ewma reward: 267.86    Epsilon: 0.010
Step: 534195    Episode: 1986   Length: 211    Total reward: 271.38    Ewma reward: 268.04    Epsilon: 0.010
Step: 534406    Episode: 1987   Length: 211    Total reward: 274.44    Ewma reward: 268.36    Epsilon: 0.010
Step: 534611    Episode: 1988   Length: 205    Total reward: 298.93    Ewma reward: 269.88    Epsilon: 0.010
Step: 534797    Episode: 1989   Length: 186    Total reward: 291.35    Ewma reward: 270.96    Epsilon: 0.010
Step: 535020    Episode: 1990   Length: 223    Total reward: 290.43    Ewma reward: 271.93    Epsilon: 0.010
Step: 535221    Episode: 1991   Length: 201    Total reward: 266.62    Ewma reward: 271.67    Epsilon: 0.010
Step: 535404    Episode: 1992   Length: 183    Total reward: 272.26    Ewma reward: 271.70    Epsilon: 0.010
Step: 535600    Episode: 1993   Length: 196    Total reward: 306.81    Ewma reward: 273.45    Epsilon: 0.010
Step: 535790    Episode: 1994   Length: 190    Total reward: 264.62    Ewma reward: 273.01    Epsilon: 0.010
Step: 536421    Episode: 1995   Length: 631    Total reward: 242.20    Ewma reward: 271.47    Epsilon: 0.010
Step: 536621    Episode: 1996   Length: 200    Total reward: 290.19    Ewma reward: 272.40    Epsilon: 0.010
Step: 536824    Episode: 1997   Length: 203    Total reward: 273.22    Ewma reward: 272.45    Epsilon: 0.010
Step: 537032    Episode: 1998   Length: 208    Total reward: 266.90    Ewma reward: 272.17    Epsilon: 0.010
Step: 537275    Episode: 1999   Length: 243    Total reward: 312.76    Ewma reward: 274.20    Epsilon: 0.010
Start Testing
total reward: 249.31
total reward: 289.95
total reward: 278.94
total reward: 277.21
total reward: 318.31
total reward: 266.27
total reward: 306.73
total reward: 301.82
total reward: 317.72
total reward: 291.27
Average Reward 289.7509202940418
(pytorch) jackkuo@lab708-Default-string:~/DLPhw6$
```

DDPG: Average Reward: 282.36

```
Step: 891533    Episode: 2977   Length: 202    Total reward: 282.89    Ewma reward: 273.98
Step: 891713    Episode: 2978   Length: 180    Total reward: 248.06    Ewma reward: 272.68
Step: 891886    Episode: 2979   Length: 173    Total reward: 261.66    Ewma reward: 272.13
Step: 892082    Episode: 2980   Length: 196    Total reward: 263.54    Ewma reward: 271.70
Step: 892275    Episode: 2981   Length: 193    Total reward: 279.98    Ewma reward: 272.11
Step: 892469    Episode: 2982   Length: 194    Total reward: 282.47    Ewma reward: 272.63
Step: 892636    Episode: 2983   Length: 167    Total reward: 264.61    Ewma reward: 272.23
Step: 892848    Episode: 2984   Length: 212    Total reward: 291.23    Ewma reward: 273.18
Step: 893045    Episode: 2985   Length: 197    Total reward: 253.63    Ewma reward: 272.20
Step: 893242    Episode: 2986   Length: 197    Total reward: 278.86    Ewma reward: 272.54
Step: 893450    Episode: 2987   Length: 208    Total reward: 271.23    Ewma reward: 272.47
Step: 893870    Episode: 2988   Length: 420    Total reward: 289.62    Ewma reward: 273.33
Step: 894073    Episode: 2989   Length: 203    Total reward: 265.83    Ewma reward: 272.95
Step: 894278    Episode: 2990   Length: 205    Total reward: 299.00    Ewma reward: 274.26
Step: 894508    Episode: 2991   Length: 230    Total reward: 293.64    Ewma reward: 275.22
Step: 894675    Episode: 2992   Length: 167    Total reward: 268.60    Ewma reward: 274.89
Step: 894870    Episode: 2993   Length: 195    Total reward: 274.07    Ewma reward: 274.85
Step: 895077    Episode: 2994   Length: 207    Total reward: 304.46    Ewma reward: 276.33
Step: 895328    Episode: 2995   Length: 251    Total reward: 292.37    Ewma reward: 277.13
Step: 895500    Episode: 2996   Length: 172    Total reward: 285.88    Ewma reward: 277.57
Step: 895706    Episode: 2997   Length: 206    Total reward: 277.85    Ewma reward: 277.59
Step: 895931    Episode: 2998   Length: 225    Total reward: 277.64    Ewma reward: 277.59
Step: 896082    Episode: 2999   Length: 151    Total reward: 270.53    Ewma reward: 277.24
Start Testing
total reward: 252.81
total reward: 285.40
total reward: 281.74
total reward: 280.37
total reward: 307.27
total reward: 268.19
total reward: 283.82
total reward: 295.76
total reward: 290.24
total reward: 278.04
Average Reward 282.3637702581553
(pytorch) jackkuo@lab708-Default-string:~/DLPhw6$
```