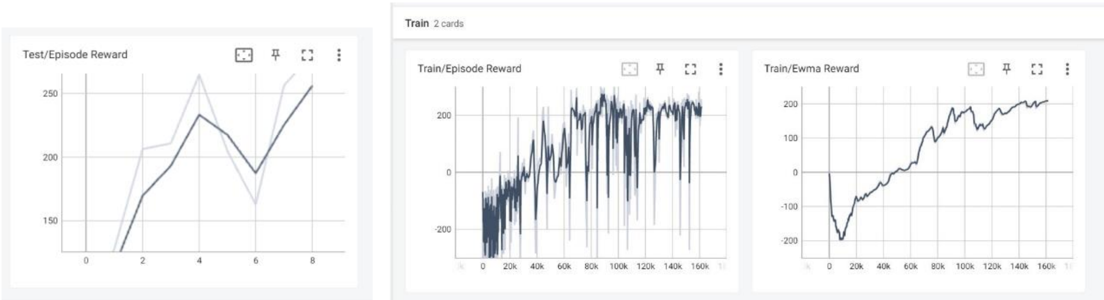
NYCU DL Lab5 – Deep Q-Network and Deep Deterministic Policy Gradient

312551086 張紀睿

Experimental Results

LunarLander-v2

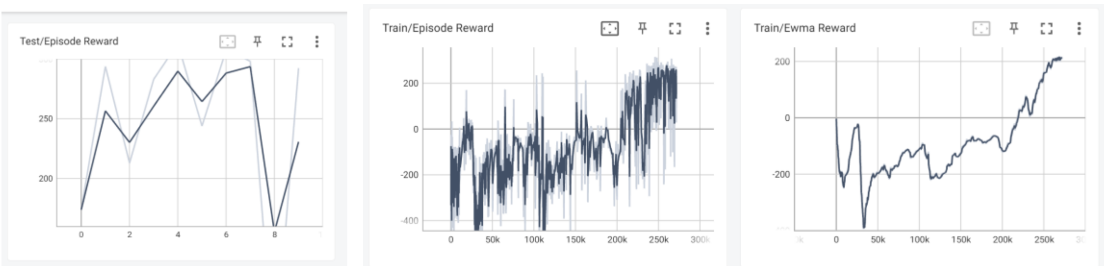
DQN




```
pp037@ec037:~/DLP_2023/Lab5$ python3 dqn.py --test_only --model best_score/DQN_LunarLander.pth
/home/pp037/.local/lib/python3.8/site-packages/gym/logger.py:30: UserWarning: WARN: Box bound precision lowered by casting to float32
  warnings.warn(colorize('%s: %s'%('WARN', msg % args), 'yellow'))
Start Testing
Episode: 1      : 81.76
Episode: 2      : 127.70
Episode: 3      : 206.26
Episode: 4      : 210.79
Episode: 5      : 264.76
Episode: 6      : 204.68
Episode: 7      : 162.78
Episode: 8      : 256.30
Episode: 9      : 281.26
Episode: 10     : 256.89
Average Reward 205.31899588858687
```

LunarLanderContinuous-v2

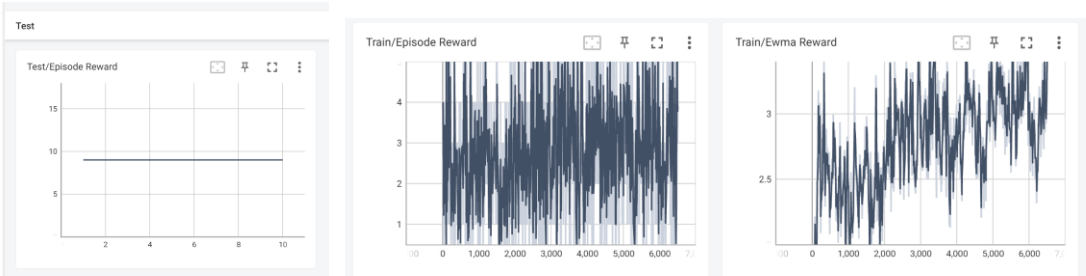
DDPG




```
pp037@ec037:~/DLP_2023/Lab5$ python3 ddpg-example.py --test_only --model models/ddpg729.pth
/home/pp037/.local/lib/python3.8/site-packages/gym/logger.py:30: UserWarning: WARN: Box bound precision lowered by casting to float32
  warnings.warn(colorize('%s: %s'%('WARN', msg % args), 'yellow'))
Start Testing
Episode: 1      : 173.76
Episode: 2      : 293.76
Episode: 3      : 212.85
Episode: 4      : 282.93
Episode: 5      : 312.80
Episode: 6      : 243.81
Episode: 7      : 307.44
Episode: 8      : 298.02
Episode: 9      : 42.12
Episode: 10     : 292.28
Average Reward 245.97685480984137
```

BreakoutNoFrameskip-v4

DQN breakout




```
pp037@ec037:~/DLP_2023/Lab5$ python3 dqn_breakout_example.py --test_only --test_model_path ckpt/dqn2_49999.pt
Start Testing
episode 1: 9.00
episode 2: 9.00
episode 3: 9.00
episode 4: 9.00
episode 5: 9.00
episode 6: 9.00
episode 7: 9.00
episode 8: 9.00
episode 9: 9.00
episode 10: 9.00
Average Reward: 9.00
```

Bonus

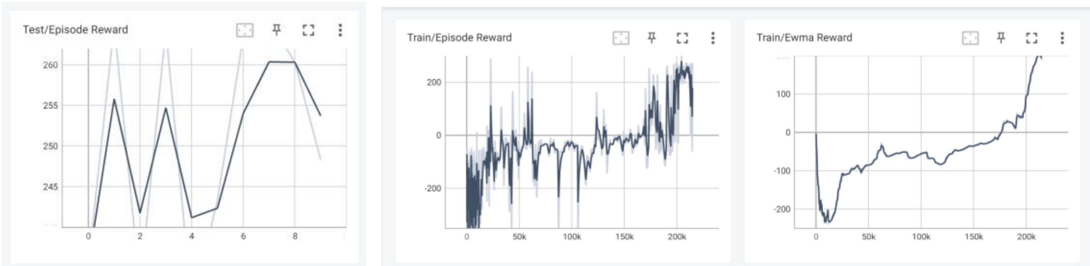
LunarLander-v2

DDQN


```
pp037@ec037:~/DLP_2023/Lab5$ python3 ddqn.py --test_only --model models/396.pth
/home/pp037/.local/lib/python3.8/site-packages/gym/logger.py:30: UserWarning: WARN: Box bound precision lowered by casting to float32
  warnings.warn(colorize('%s: %s'%('WARN', msg % args), 'yellow'))
Start Testing
Episode: 1      : 148.48
Episode: 2      : 204.24
Episode: 3      : 163.04
Episode: 4      : 170.96
Episode: 5      : 259.59
Episode: 6      : 168.55
Episode: 7      : 232.17
Episode: 8      : 229.51
Episode: 9      : 238.99
Episode: 10     : 223.28
Average Reward 203.87958826469014
```

LunarLanderContinuous-v2

TD3




```
pp037@ec037:~/DLP_2023/Lab5$ python3 td3.py --test_only --model models/TD3365.pth
/home/pp037/.local/lib/python3.8/site-packages/gym/logger.py:30: UserWarning: WARN: Box bound precision lowered by casting to float32
  warnings.warn(colorize('%s: %s'%('WARN', msg % args), 'yellow'))
Start Testing
Episode: 1      : 235.45
Episode: 2      : 264.97
Episode: 3      : 232.54
Episode: 4      : 264.38
Episode: 5      : 230.54
Episode: 6      : 243.26
Episode: 7      : 263.49
Episode: 8      : 265.54
Episode: 9      : 260.28
Episode: 10     : 248.33
Average Reward 250.87718375758746
```

Questions

Describe your major implementation of both DQN and DDPG in detail:

(1) Your implementation of Q network updating in DQN

```python
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 ##
    state = torch.Tensor(state).to(self.device)
    q_value = self._behavior_net(state).gather(1, action.long())
    with torch.no_grad():
        output = self._target_net(next_state)
        q_next, _ = torch.max(output, 1)
        q_next = q_next.view(-1, 1)
        q_target = reward + q_next * gamma * (1 - done)
    criterion = nn.MSELoss()
    loss = criterion(q_value, q_target)
    #raise NotImplementedError
    # optimize
    self._optimizer.zero_grad()
    loss.backward()
    nn.utils.clip_grad_norm_(self._behavior_net.parameters(), 5)
    self._optimizer.step()
```

先從 memory 中進行 sample，取得 state, action, reward, next_state 與 done。將 state 送入 behavior_net 並根據行動取得 Q 值，再使用 target net 用 next_state 預測未來的 q 值，乘上權重並與當前 reward 相加以得到 target Q value，將 q value 與 target q value 計算 MSELoss 後用以更新 behavior_net。

(2) Your implementation and the gradient of actor updating in DDPG.

$$\nabla_{\theta_\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s_i}$$

```
action = actor_net(state)
actor_loss = -critic_net(state, action).mean()
```

由 actor_net 預測出 action 後，將 state 與 action 輸入 Critic net 取得 Q 值並將其取 mean 以得到 policy gradient，用以更新 actor network。

(3) Your implementation and the gradient of critic updating in DDPG.

```
q_value = critic_net(state, action)
with torch.no_grad():
    a_next = target_actor_net(next_state)
    #print(next_state.shape)
    #print(a_next.shape)
    q_next = target_critic_net(next_state, a_next)
    q_target = reward + q_next * gamma * (1 - done)
criterion = nn.MSELoss()
critic_loss = criterion(q_value, q_target)
```

將 sample 出來的 state 與 action 輸入 critic net 預測出 Q value。之後，將 next_state 輸入 target actor net 以得到下一個 action，再由 target critic net 根據 next_state 與 next action 取得未來的 Q 值後，乘上權重並與當前 reward 相加以得到 target Q value，將 q value 與 target q value 計算 MSELoss 後用以更新 critic net。

Explain effects of the discount factor

Discount factor 會影響模型對未來的 reward 的重視程度，discount factor 越小，則模型越重視即時的獎勵，而 discount factor 越大則會使模型對未來獎勵的重視程度越大。

Explain benefits of epsilon-greedy in comparison to greedy action selection

Greedy 會讓模型選擇最高 Q 值的 action，而 Epsilon-greedy 會使模型有一定機率隨機選擇 action，這使模型有更多 explore 的機會，以避免掉入 local optima，並提升整體學習的機會。

Explain the necessity of the target network

使用 target network 可以讓訓練變得更穩定，避免在模型進行監督的同時訓練並更新模型，改變了模型訓練的目標造成模型無法收斂。

Describe the tricks you used in Breakout and their effects, and how they differ from those used in LunarLander

在 Breakout 中，將四個 frame 結合在一起後做為一個 state，再輸入 Network 進行預測，以此獲得 frame 與 frame 之動態關係，與 LunarLander 僅使用當前的 state 不同，這是因為在 breakout 中，即使畫面相同，也會因為球的移動方向造成需要的行為有所不同，因此會需要將多個 frame stack 在一起。