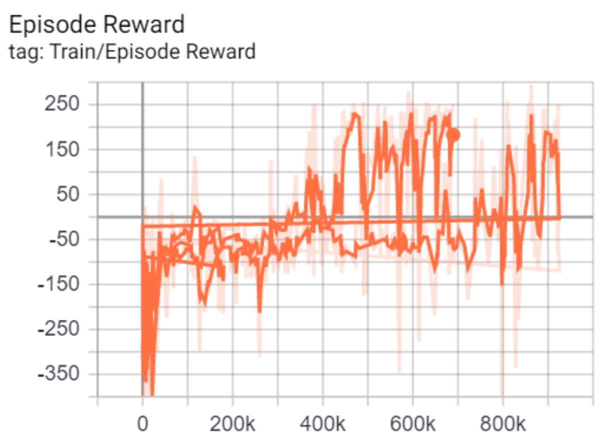
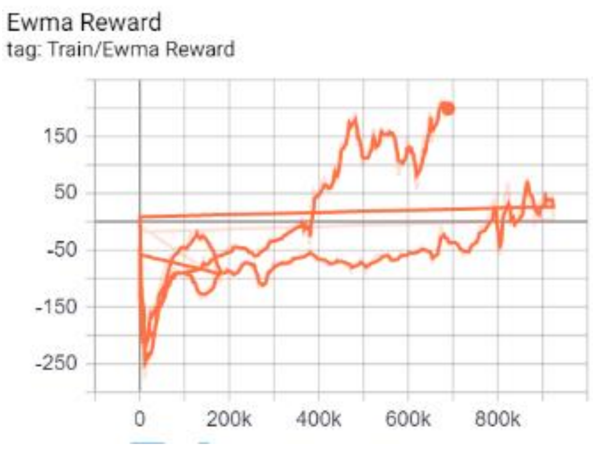
Lab 6: Deep Q-Network and Deep Deterministic Policy Gradient

309552042 網工所 黃敏涓

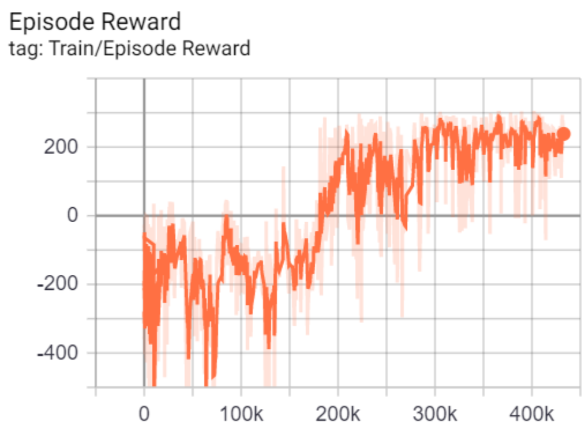
I. Report (80%)

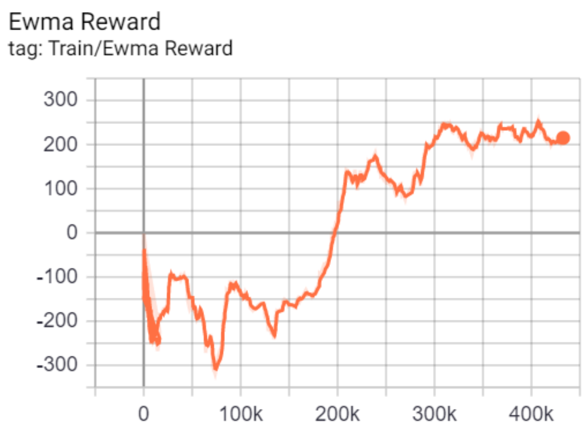
1. A tensorboard plot shows episode rewards of at least 800 training episodes in LunarLander-v2 (5%)





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

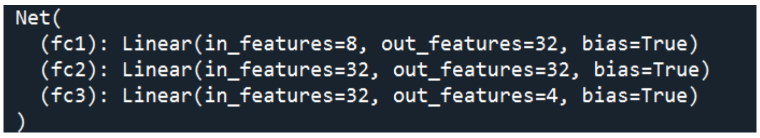




3. Describe your major implementation of both algorithms in detail. (20%)

DQN:

-**Network Architecture**



```
Net(
  (fc1): Linear(in_features=8, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=32, bias=True)
  (fc3): Linear(in_features=32, out_features=4, bias=True)
)
```

每次 action selection 後，做 Experience Replay，以構建一個儲存把樣本都儲存下來，通過隨機取樣去除相關性，接著透過 perform gradient descent on $Loss = (r + \gamma max_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2$ 以更新 Q network 權重，每隔一定的步數再將 Q network 的權重更新到 target network(獨立且慢於當前 Q-Network 來計算 y，使得訓練震盪發散可能性降低，更加穩定)。

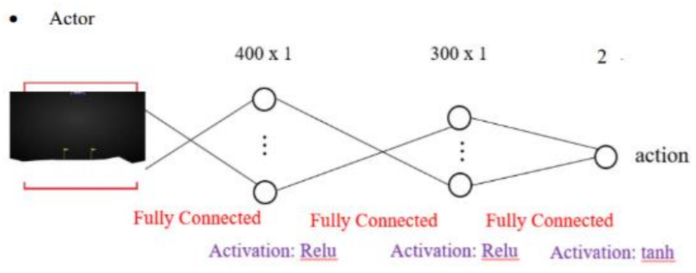
DDPG:

Select action $\boxed{a_t = \mu(s_t|\theta^\mu) + N_t}$ according to the current policy and exploration noise

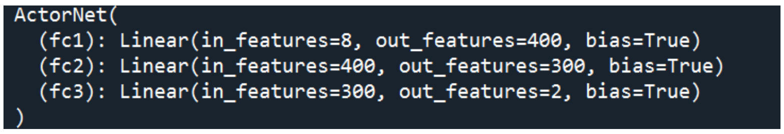
In continuous action spaces, exploration is done via adding noise to the action itself.

- Actor Net Architecture:

Based on



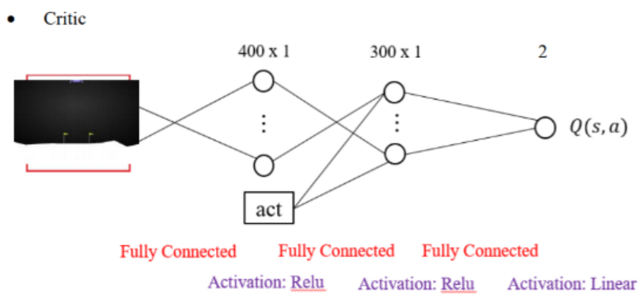
由三層 fully-connected layer 所構成



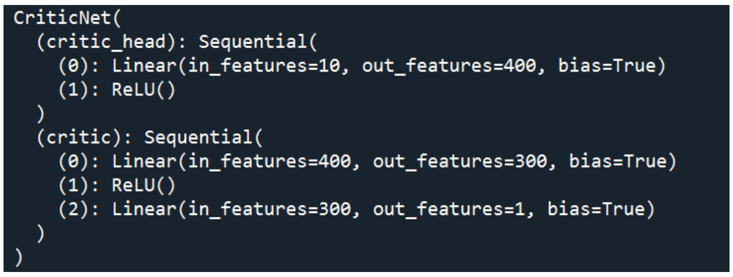
```
ActorNet(
  (fc1): Linear(in_features=8, out_features=400, bias=True)
  (fc2): Linear(in_features=400, out_features=300, bias=True)
  (fc3): Linear(in_features=300, out_features=2, bias=True)
)
```

-Critic Net Architecture:

Based on



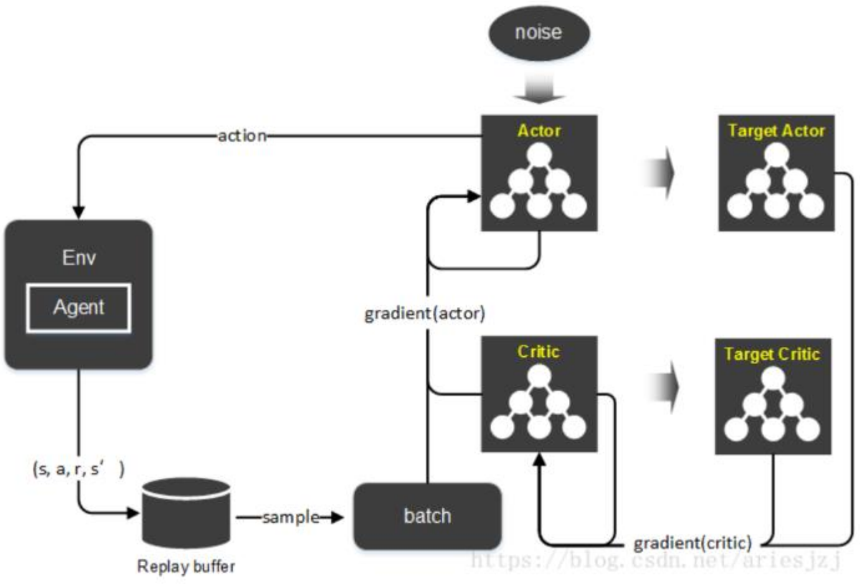
由三層 fully-connected layer 所構成，最後一層的 activation function 改為 linear


```
CriticNet(
  (critic_head): Sequential(
    (0): Linear(in_features=10, out_features=400, bias=True)
    (1): ReLU()
  )
  (critic): Sequential(
    (0): Linear(in_features=400, out_features=300, bias=True)
    (1): ReLU()
    (2): Linear(in_features=300, out_features=1, bias=True)
  )
)
```

DDPG 分別爲 actor 和 critic 各創建兩個神經網絡，在訓練完一個 mini-batch 的數據之後，通過 SGD 算法更新 actor_net, critic_net 的參數，然後再通過 soft update 算法更新 target_actor_net , target_critic_net 的參數。

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{\mu'}$$

soft update 是一種 running average 的算法，因 target 網絡參數變化小，用於在訓練過程中計算網絡的 gradient，比較穩定，使訓練易於收斂。
整個演算法運作流程如圖:



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

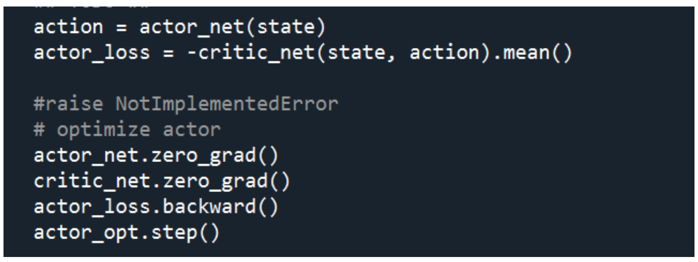
In the DDPG algorithm, the authors use *Ornstein-Uhlenbeck Process* to add noise to the action output (Uhlenbeck & Ornstein, 1930):

The *Ornstein-Uhlenbeck Process* generates noise that is correlated with the previous noise, as to prevent the noise from canceling out or “freezing” the overall dynamics, here, instead, we simply add Gaussian noise to the action output.

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

Since we are updating the policy in an off-policy way with batches of experience, we take the mean of the sum of gradients calculated from the mini-batch to upgrade actor network:

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



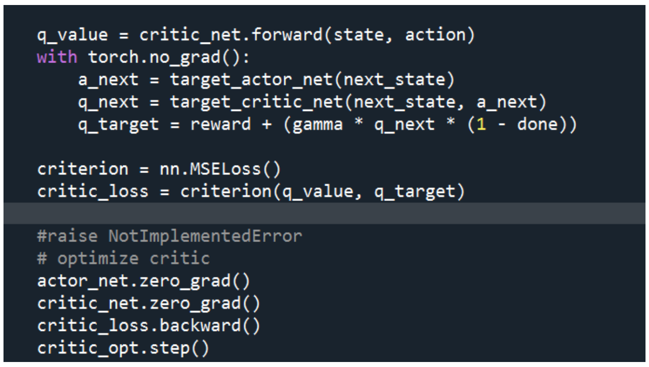
```
action = actor_net(state)
actor_loss = -critic_net(state, action).mean()

#raise NotImplementedError
# optimize actor
actor_net.zero_grad()
critic_net.zero_grad()
actor_loss.backward()
actor_opt.step()
```

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

根據 critic 的損失函式來更新 crtic 網路:

Update critic by minimizing the loss: $L = \frac{1}{N}\Sigma_i\left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$



```
q_value = critic_net.forward(state, action)
with torch.no_grad():
    a_next = target_actor_net(next_state)
    q_next = 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)

#raise NotImplementedError
# optimize critic
actor_net.zero_grad()
critic_net.zero_grad()
critic_loss.backward()
critic_opt.step()
```

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

A discount factor will result in state/action values representing the immediate reward, while a higher discount factor will result in the values representing the cumulative discounted future reward an agent expects to receive (behaving under a given policy).

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

Greedy action selection 會有 Exploration-Exploitation Dilemma 的問題，導致無法找到 Action value 最大的選項，而 Epsilon-greedy 為簡單處理 Exploration-Exploitation Trade-Off 的方法，每一回合隨機選擇一個 0~1 的數字，如果該數字小於 ε(0~1)，則選擇 Explore，反之，選擇具有最大 Action Value 的選項。

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

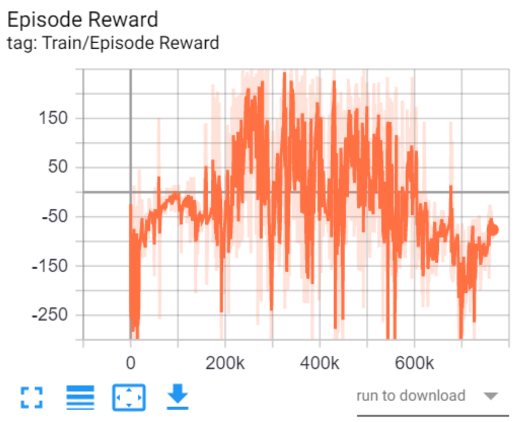
使得在一段時間裡目標 Q 值使保持不變，一定程度降低了當前 Q 值和目標 Q 值的相關性，提高了算法穩定性。

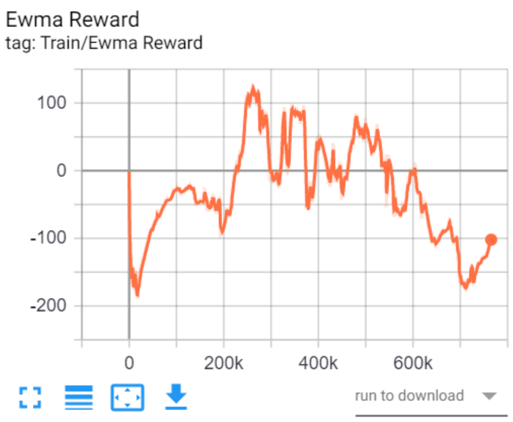
10. Explain the effect of replay buffer size in case of too large or too small. (5%)

The larger the experience replay, the less likely you will sample correlated elements, hence the more stable the training of the NN will be. However, a large experience replay also requires a lot of memory and it might slow training. So, there is a trade-off between training stability (of the NN) and memory requirements.

II. Report Bonus (20%)

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





2.Extra hyperparameter tuning, e.g., Population Based Training. (10%)

III. Performance (20%)

[LunarLander-v2] Average reward of 10 testing episodes: Average ÷ 30

244.3713÷30 = 8.14

[LunarLanderContinuous-v2] Average reward of 10 testing episodes: Average ÷ 30

200.3683÷30= 6.678