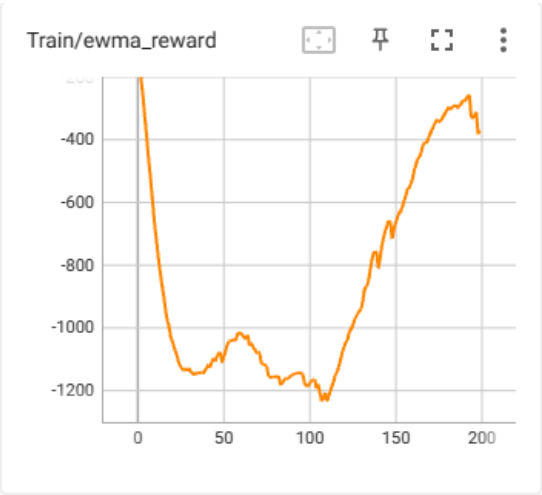
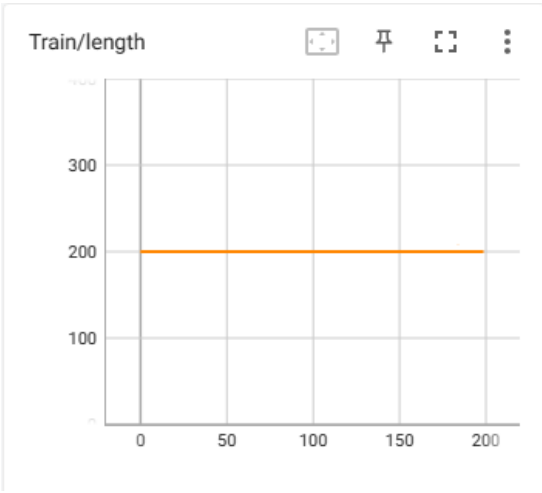
RL HW2 Report

311551059 陳昱丞

(a) Pendulum

- Results
  - In task “Pendulum-v0”, it is likely to achieve an approximate reward of 0 in around 200 episodes.





- Hyperparameters
  - Learning rate: lr_a=1e-4, lr_c=1e-3
  - Batch size: 128
  - NN architecture:

```python
class Actor(nn.Module):
    def __init__(self, hidden_size, num_inputs, action_space):
        super(Actor, self).__init__()
        self.action_space = action_space
        num_outputs = action_space.shape[0]

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Construct your own actor network
        self.fc1 = nn.Linear(num_inputs, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, num_outputs)
        self.relu = nn.ReLU()
        self.tanh = nn.Tanh()
        ########## END OF YOUR CODE ##########

    def forward(self, inputs):

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Define the forward pass your actor network
        x = self.fc1(inputs)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        x = self.tanh(x)
        return x
        ########## END OF YOUR CODE ##########
```

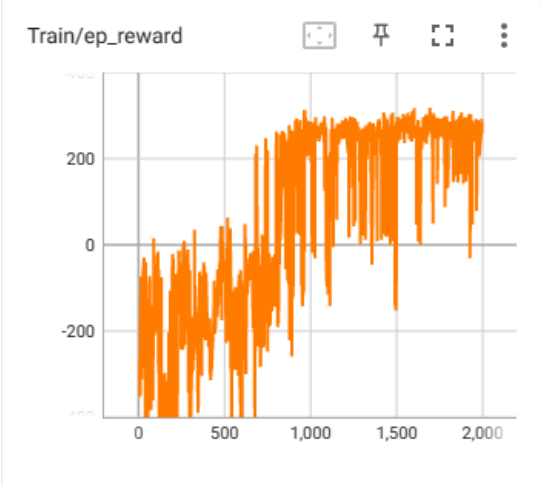
```python
class Critic(nn.Module):
    def __init__(self, hidden_size, num_inputs, action_space):
        super(Critic, self).__init__()
        self.action_space = action_space
        num_outputs = action_space.shape[0]

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Construct your own critic network
        self.fc1 = nn.Linear(num_inputs + num_outputs, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, 1)
        self.relu = nn.ReLU()
        ########## END OF YOUR CODE ##########

    def forward(self, inputs, actions):
        ########## YOUR CODE HERE (5~10 lines) ##########
        # Define the forward pass your critic network
        x = self.fc1(torch.cat([inputs, actions], dim=1))
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        return x
        ########## END OF YOUR CODE ##########
```

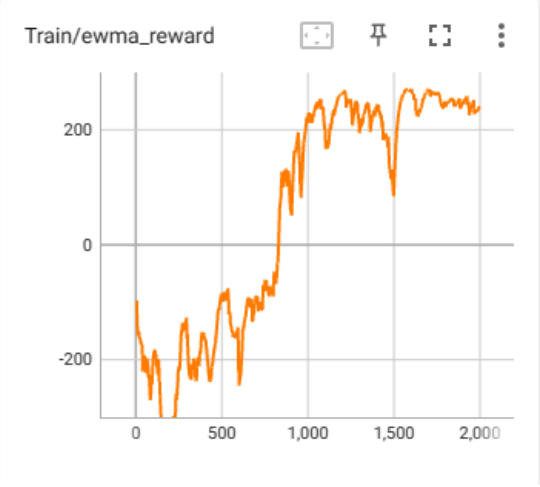
(hidden_size = 128)

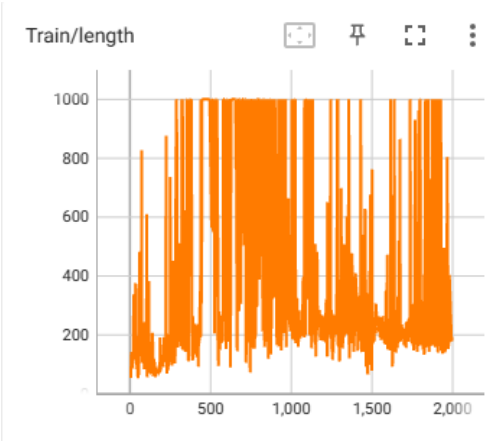
(b) LunarLanderContinuous

- Results
  - In task “LunarLanderContinuous-v2”, it can achieve reward > 200 in around 2000 episodes.

```
Episode: 1980, length: 189, reward: 273.04, ewma reward: 230.42
Episode: 1981, length: 170, reward: 260.10, ewma reward: 231.90
Episode: 1982, length: 189, reward: 282.77, ewma reward: 234.44
Episode: 1983, length: 404, reward: 205.78, ewma reward: 233.01
Episode: 1984, length: 167, reward: 251.03, ewma reward: 233.91
Episode: 1985, length: 1000, reward: 169.36, ewma reward: 230.68
Episode: 1986, length: 178, reward: 246.18, ewma reward: 231.46
Episode: 1987, length: 1000, reward: 147.09, ewma reward: 227.24
Episode: 1988, length: 217, reward: 263.60, ewma reward: 229.06
Episode: 1989, length: 175, reward: 260.08, ewma reward: 230.61
Episode: 1990, length: 1000, reward: 121.13, ewma reward: 225.14
Episode: 1991, length: 467, reward: 254.62, ewma reward: 226.61
Episode: 1992, length: 275, reward: 277.62, ewma reward: 229.16
Episode: 1993, length: 224, reward: 264.79, ewma reward: 230.94
Episode: 1994, length: 158, reward: 286.02, ewma reward: 233.70
Episode: 1995, length: 197, reward: 258.41, ewma reward: 234.93
Episode: 1996, length: 191, reward: 247.61, ewma reward: 235.56
Episode: 1997, length: 191, reward: 291.95, ewma reward: 238.38
Episode: 1998, length: 191, reward: 283.73, ewma reward: 240.65
Episode: 1999, length: 178, reward: 257.08, ewma reward: 241.47
Saving models to preTrained/ddpg_actor_LunarLanderContinuous-v2_04202023_082833_.pth and preTrained/ddpg_critic_LunarLanderContinuous-v2_04202023_082833_.pth
```







- Hyperparameters
  - Learning rate: lr_a=1e-3, lr_c=1e-3
  - Batch size: 64
  - NN architecture:

```python
class Actor(nn.Module):
    def __init__(self, hidden_size, num_inputs, action_space):
        super(Actor, self).__init__()
        self.action_space = action_space
        num_outputs = action_space.shape[0]

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Construct your own actor network
        self.fc1 = nn.Linear(num_inputs, hidden_size[0])
        self.fc2 = nn.Linear(hidden_size[0], hidden_size[1])
        self.fc3 = nn.Linear(hidden_size[1], num_outputs)
        self.relu = nn.ReLU()
        self.tanh = nn.Tanh()
        ########## END OF YOUR CODE ##########

    def forward(self, inputs):

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Define the forward pass your actor network
        x = self.fc1(inputs)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        x = self.tanh(x)
        return x
        ########## END OF YOUR CODE ##########
```

```python
class Critic(nn.Module):
    def __init__(self, hidden_size, num_inputs, action_space):
        super(Critic, self).__init__()
        self.action_space = action_space
        num_outputs = action_space.shape[0]

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Construct your own critic network
        self.fc1 = nn.Linear(num_inputs + num_outputs, hidden_size[0])
        self.fc2 = nn.Linear(hidden_size[0], hidden_size[1])
        self.fc3 = nn.Linear(hidden_size[1], 1)
        self.relu = nn.ReLU()
        ########## END OF YOUR CODE ##########

    def forward(self, inputs, actions):
        ########## YOUR CODE HERE (5~10 lines) ##########
        # Define the forward pass your critic network
        x = self.fc1(torch.cat([inputs, actions], dim=1))
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        return x
        ########## END OF YOUR CODE ##########
```

(hidden_size = 400, 300)