Report of DDPG for Continuous Control

1. Implementation and model architectures

We employ the DDPG algorithm with randomly sample experience replay:

Algorithm 1 Deep Deterministic Policy Gradient

Input: initial policy parameters θ, Q-function parameters ϕ, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for however many updates **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

      Update Q-function by one step of gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\phi(s, a) - y(r, s', d)\right)^2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

      Update target networks with

$$\begin{aligned} \phi_{\text{targ}} &\leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi \\ \theta_{\text{targ}} &\leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta \end{aligned}$$

    end for
  end if
until convergence

1.1 Architecture of the ***Actor network***

We use a three-layer fully connected network as the surrogate of the Actor. The first and second layers are applied the leaky Relu activation function, while the last layer links with tanh activation function.The network is sketched as follows:

```
class Actor(nn.Module):
    """Actor in DDPG Model."""

    def __init__(self, state_size, action_size, seed):
        """Initialize parameters and build model.
        Params
        ======
            state_size (int): Dimension of each state
            action_size (int): Dimension of each action
            seed (int): Random seed
        """
        super(Actor, self).__init__()
        self.seed = torch.manual_seed(seed)

        # linear layers
        self.fc1 = nn.Linear(state_size,512)
        self.fc2 = nn.Linear(512,256)
        self.fc3 = nn.Linear(256,action_size)

        # activation functions
        self.tanh = nn.Tanh()

    def forward(self, state):
        """Build a network that maps state -> action values."""

        x = self.fc1(state)
        x = F.leaky_relu(x)
        x = self.fc2(x)
        x = F.leaky_relu(x)
        x = self.fc3(x)
        x = self.tanh(x)
        return x
```

1.2 Architecture of the ***Critic network***

We also use a three-layer fully connected network as the surrogate of the Critic network. The first and second layers are applied the leaky Relu activation function, while the last layer links with no activation function.The network is sketched as follows:

```python
class Critic(nn.Module):
    """Critic in DDPG Model."""

    def __init__(self, state_size, action_size, seed):
        """Initialize parameters and build model.
        Params
        ======
            state_size (int): Dimension of each state
            action_size (int): Dimension of each action
            seed (int): Random seed
        """
        super(Critic, self).__init__()
        self.seed = torch.manual_seed(seed)

        # linear layers
        self.fc1 = nn.Linear(state_size,512)
        self.fc2 = nn.Linear(512+action_size,256)
        self.fc3 = nn.Linear(256,1)

    def forward(self, state, action):
        """Build a network that maps state-action pair into Q-values."""

        x = self.fc1(state)
        x = F.leaky_relu(x)
        x = self.fc2(torch.cat((x, action), dim=1))
        x = F.leaky_relu(x)
        x = self.fc3(x)

        return x
```

2. Parameters and results

Table 1. gives the parameters we used in the network.

Table 1. Parameters

Para.	Value	Para.	Value
BUFFER_SIZE	10^6	BATCH_SIZE	128
LR	1×10^{-3}	UPDATE_EVERY	20
GAMMA	0.99	TAU	1×10^{-3}
eps_start	1.0	eps_end	0.1
eps_decay	0.99	NUM_LEARNING	10

The following list the average reward per 100 episodes. With the above parameters, the agent is able to receive an average reward over 30 in 428 episodes.

Episode 10 Average Score: 0.95 Score: 1.05
Episode 20 Average Score: 0.91 Score: 1.44
Episode 30 Average Score: 1.09 Score: 0.90
Episode 40 Average Score: 1.33 Score: 0.58
Episode 50 Average Score: 1.67 Score: 2.81
Episode 60 Average Score: 2.08 Score: 4.15
Episode 70 Average Score: 2.84 Score: 7.060
Episode 80 Average Score: 3.73 Score: 11.53
Episode 90 Average Score: 4.61 Score: 25.77
Episode 100 Average Score: 5.62 Score: 16.76
Episode 110 Average Score: 7.33 Score: 20.31
Episode 120 Average Score: 9.37 Score: 24.22
Episode 130 Average Score: 11.39 Score: 22.02
Episode 140 Average Score: 13.40 Score: 25.00
Episode 150 Average Score: 15.73 Score: 24.49
Episode 160 Average Score: 17.70 Score: 18.61
Episode 170 Average Score: 19.59 Score: 27.28
Episode 180 Average Score: 20.80 Score: 25.88
Episode 190 Average Score: 22.21 Score: 23.31
Episode 200 Average Score: 23.02 Score: 26.91
Episode 210 Average Score: 23.61 Score: 27.05
Episode 220 Average Score: 23.92 Score: 24.15
Episode 230 Average Score: 24.19 Score: 12.47
Episode 240 Average Score: 24.27 Score: 20.08
Episode 250 Average Score: 24.48 Score: 32.62
Episode 260 Average Score: 25.10 Score: 35.48
Episode 270 Average Score: 25.26 Score: 23.58
Episode 280 Average Score: 25.54 Score: 28.44
Episode 290 Average Score: 25.68 Score: 27.99
Episode 300 Average Score: 25.99 Score: 28.72
Episode 310 Average Score: 26.53 Score: 35.51
Episode 320 Average Score: 27.05 Score: 28.77
Episode 330 Average Score: 27.22 Score: 28.56
Episode 340 Average Score: 27.42 Score: 27.01
Episode 350 Average Score: 27.05 Score: 34.95
Episode 360 Average Score: 27.18 Score: 34.55
Episode 370 Average Score: 27.72 Score: 37.33
Episode 380 Average Score: 28.37 Score: 39.00
Episode 390 Average Score: 28.82 Score: 27.70
Episode 400 Average Score: 29.33 Score: 36.02
Episode 410 Average Score: 29.45 Score: 33.64
Episode 420 Average Score: 29.62 Score: 32.85
Episode 428 Average Score: 30.08 Score: 34.81
Environment solved in 428 Episodes   Average Score: 30.08

Figure 1 pictures the reward per episode.

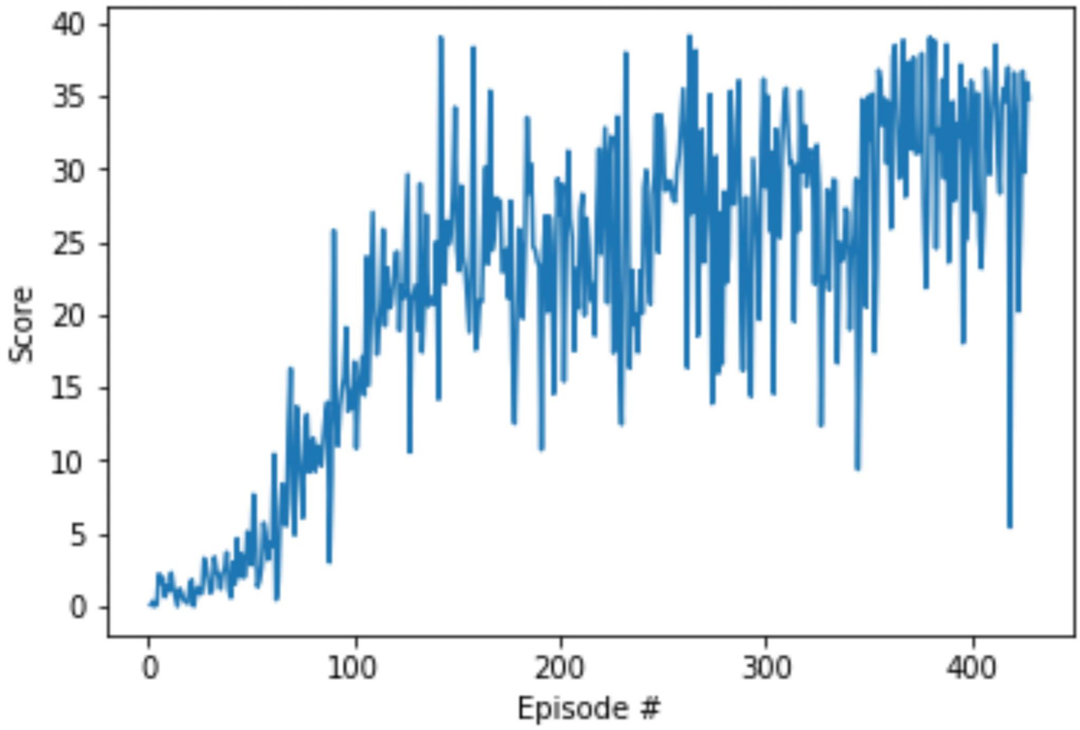

Figure 1. Reward every episode with TAU = 1e-3

3. How to improve the performance

3.1 Choose proper soft-update TAU

In the train, we first use TAU = 1e-3 and the agent gets average score over 30 at the 428-th episode. Then, we change the TAU to TAU = 5e-3 and TAU = 5e-4 and we find that the agent obtains average score over 30 at 611-th episode with larger TAU while only at 269-th episode with smaller TAU, shown in Figure 2. However, we cannot set the TAU to zero for the target and local networks will separate forever. Meanwhile, we find that with the decrease of TAU, the oscillation of the waveform decreases.

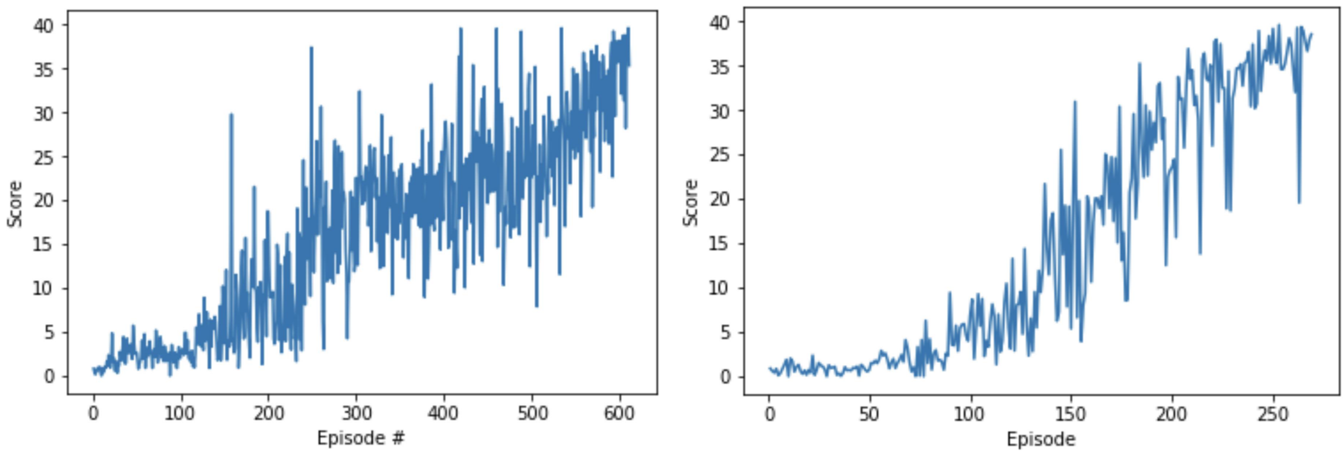

Figure 2. Reward every episode with a) TAU = 1e-3, b) TAU = 5e-4

3.2 Increase the structure of the network

We choose the number of two hidden nodes from (128*64) to (256*128) and then finally to (512*256). The first two networks fail to converge in a quick speed, while the last one with more information achieves eventually. However, when we add another layer to (512*256*128), the agent, on the contrary, cannot learn anything at this time.

```
Episode 10      Average Score: 0.05     Score: 0.00
Episode 20      Average Score: 0.03     Score: 0.00
Episode 30      Average Score: 0.02     Score: 0.00
Episode 40      Average Score: 0.02     Score: 0.00
```

This problem even occurs when we decrease the number of parameters of the network to (256*128*64). These phenomenon shows that the more complex networks do not always fit all models.

```
Episode 10      Average Score: 0.01     Score: 0.07
Episode 20      Average Score: 0.02     Score: 0.00
Episode 30      Average Score: 0.02     Score: 0.00
Episode 40      Average Score: 0.02     Score: 0.00
Episode 50      Average Score: 0.01     Score: 0.00
Episode 60      Average Score: 0.01     Score: 0.00
Episode 70      Average Score: 0.01     Score: 0.00
```