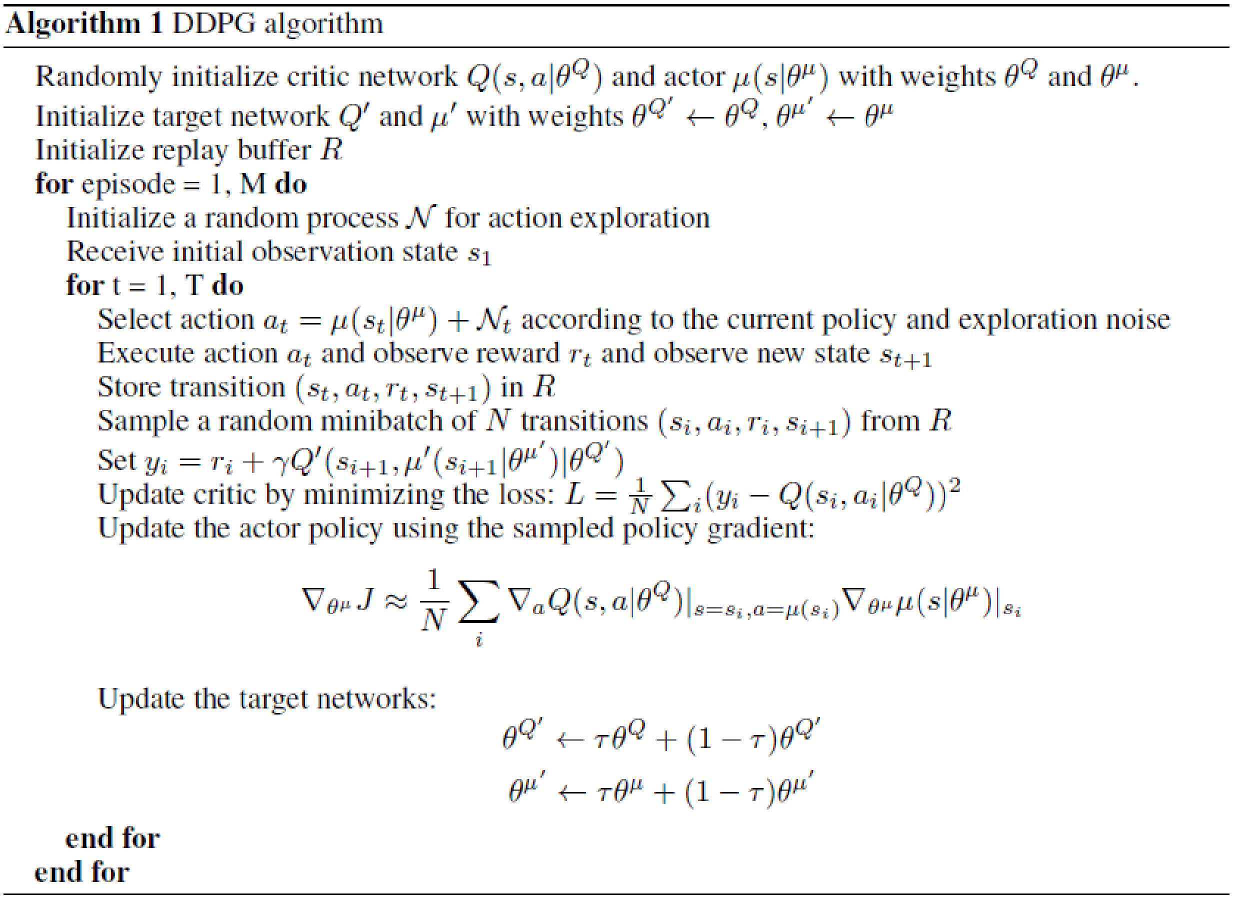
Report

Algorithm

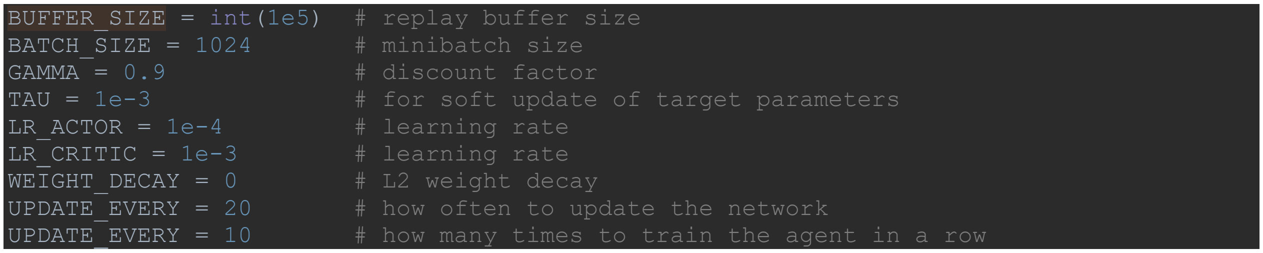
Selected Algorithm: DDPG (*ddpg_agent.py*)



Algorithm 1 DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state $s_1$
    for t = 1, T do
        Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
        Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
        Update the actor policy using the sampled policy gradient:
            $$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$
        Update the target networks:
            $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
            $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
    end for
end for
```

Parameters chosen for the DDPG Agent:



```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 1024       # minibatch size
GAMMA = 0.9             # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-4         # learning rate
LR_CRITIC = 1e-3        # learning rate
WEIGHT_DECAY = 0        # L2 weight decay
UPDATE_EVERY = 20       # how often to update the network
UPDATE_EVERY = 10       # how many times to train the agent in a row
```

The agent consists of two different NN architectures:

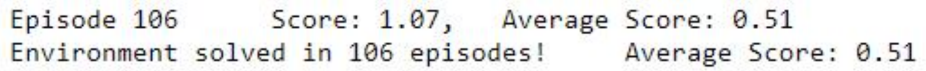
Actor

- A three-layer network with following number of units: Input (33) -> Hidden (256) -> Output (4)

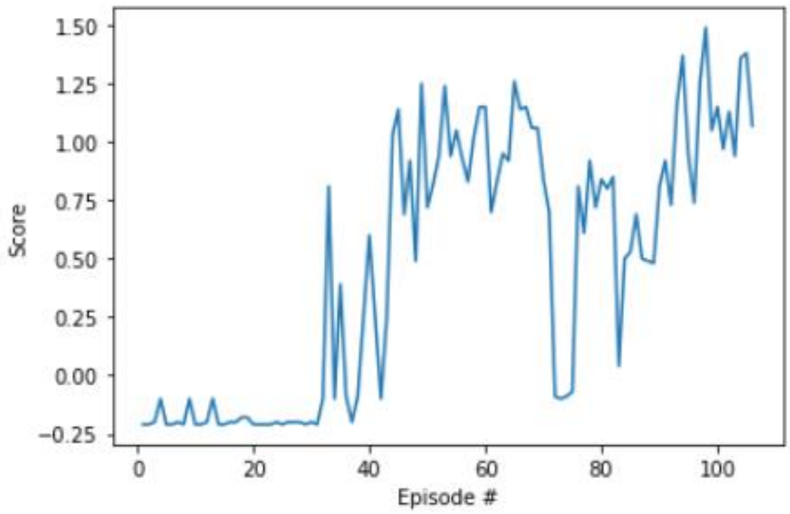
Critic

- A five-layer network with following number of units: Input (33) -> Hidden1 (256) -> Hidden2 (260) -> Hidden3 (128) -> Output (1)
-

Environment was solved in 106 episodes (as can be seen in the following chart as well as in the *Tennis.ipynb*).



```
Episode 106      Score: 1.07,   Average Score: 0.51
Environment solved in 106 episodes!     Average Score: 0.51
```



Modifications compared to the lecture

The selected method, approach and parameters are exactly same as in case of Continuous Control project. No modification has been done in order to achieve the goal.

Batch normalization added according to the DDPG paper to all the input and layers in actor and the input and all layers before the action input in the critic – here in both cases it means only once

Bacth size increased to 1024

Agent is trained always after 10 steps, but 5 times in a row

Sigma for adding noise decreased to 0.1

Improvements

Compare with other algorithms: PPO, A3C

Experiment with deeper actor network

Experiment with wider networks