1. Learning Architecture
    a) Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' 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

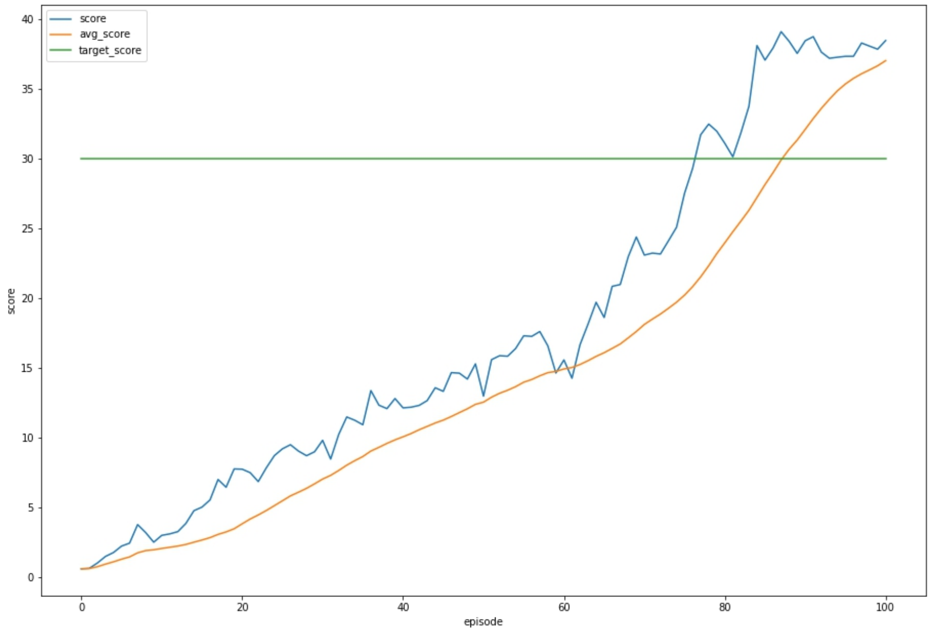
---

    b) Model Structure
        i. Actor:
            1. Input size = State size = 33
            2. Hidden layers(2)
                a) Fully connected with 128 batch-normalized rectifiers
                b) Fully connected with 256 rectifiers
            3. Output size = Action size = 4
        ii. Critic:
            1. Input 1 size = State size = 33 at 1^{st} layer
            2. Input 2 size = Action size = 4 concat to 2^{nd} layer
            3. Hidden layers(2)
                a) Fully connected with 128 batch-normalized rectifiers
                b) Fully connected with 256+4 rectifiers
            4. Output size = Value function = 1
    c) Hyperparameters

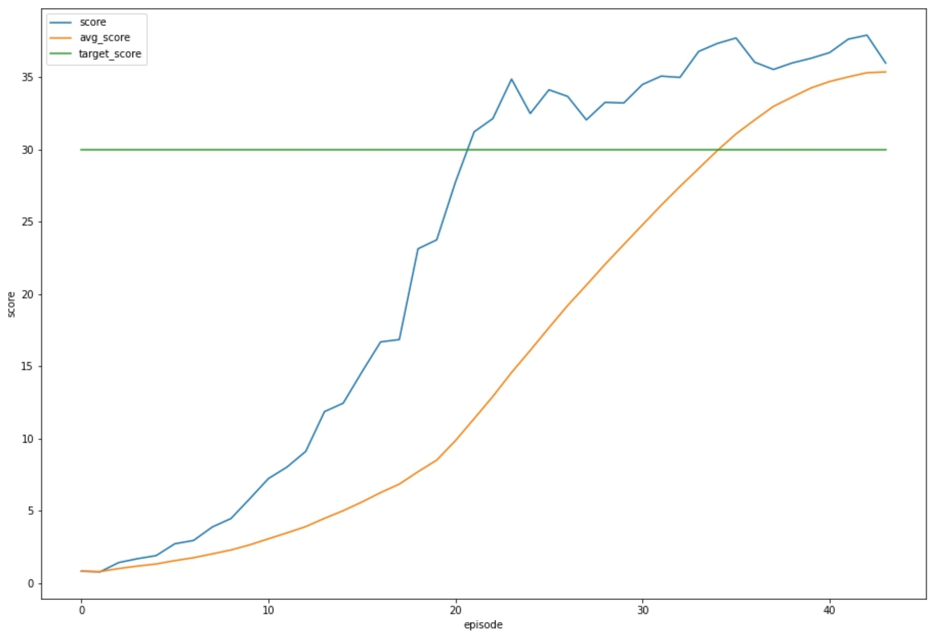
i.	Batch size	128
ii.	Memory buffer size	1e6
iii.	Number of episodes	1000
iv.	Target score	30.0
v.	Discount factor gamma	1e-3
vi.	Learning rate for Actor	1e-4
vii.	Learning rate for Critic	1e-3

viii. Update Period 20(5 for prioritized exp replay)
ix. Update Times per update 7(1 for prioritized exp replay)
x. Weight Decay 0
xi. Agent number 20
xii. Alpha(prioritized exp replay) 0.7
xiii. Beta(prioritized exp replay) 0.8

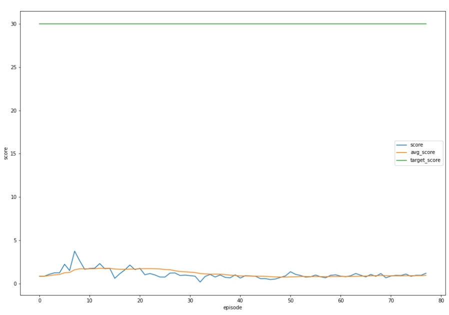
2. Results
   a) Original DDPG
      i. Folder: (ddpg_result_2022_10_04_16_14_37)
      ii. Result:

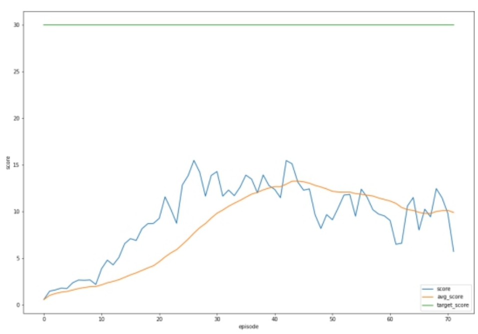


   b) DDPG with prioritized experience replay
      i. Folder: (ddpg_result_2022_10_08_21_09_12)
      ii. Result:

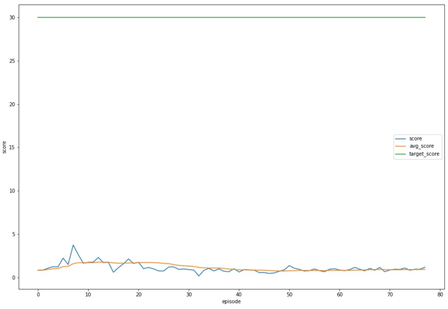


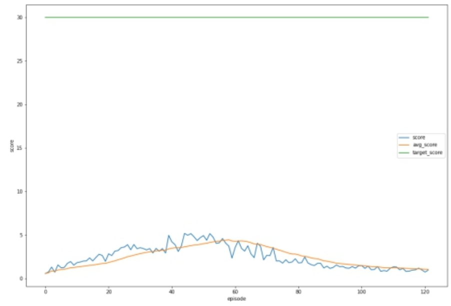
3. Conclusions
   a) Both methods converge within 100 episodes(however in the 2nd one, the training stopped early because a communication error occurred in the unity agent)
   b) Compare to original DDPG, DDPG with prioritized exp replay converge much faster(with in 40 episodes) and require much less updates(1 update per update time compared to 7 updates per update time)
   c) The training initially did not converge, so we did following steps to gradually improve the performance
      i. Increase batch size (40->128): better
         1. left (ddpg_result_2022_09_30_17_28_04)
         2. right (ddpg_result_2022_10_01_19_12_28)



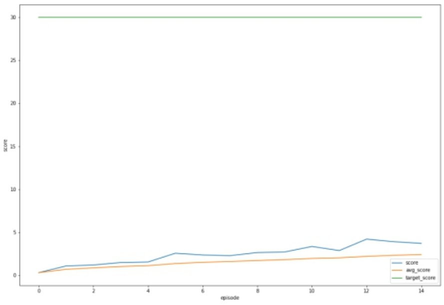


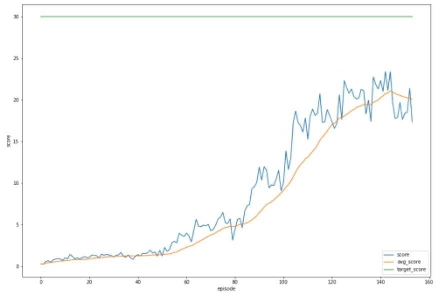
      ii. Add Batch Normalization
         1. left (ddpg_result_2022_09_30_17_28_04)
         2. right (ddpg_result_2022_10_01_19_12_28)



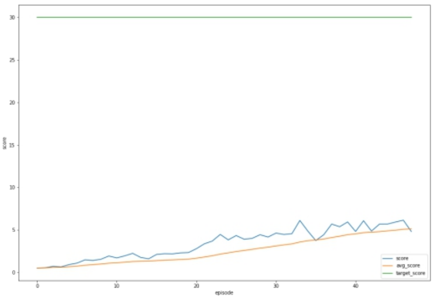


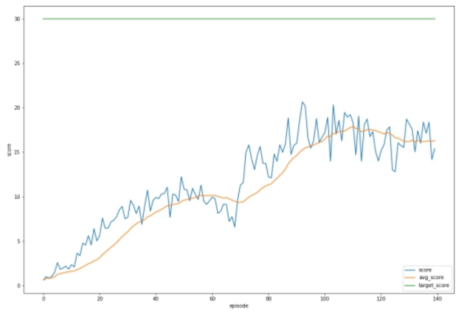
iii. Fix OUNoise error by adding -0.5 bias: better
   1. left (ddpg_result_2022_10_02_23_57_47)
   2. right (ddpg_result_2022_10_03_02_05_59)



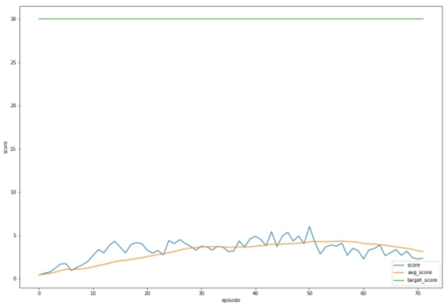


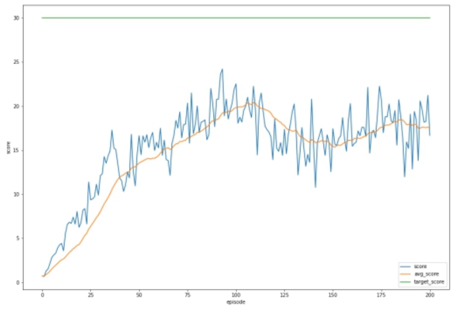
iv. Decrease noise std(0.2->0.05):better
   1. left (ddpg_result_2022_10_02_10_04_21)
   2. right (ddpg_result_2022_10_02_12_18_34)





v. Increase buffer size(1e5->1e6): better
   1. left (ddpg_result_2022_10_01_22_51_33)
   2. right (ddpg_result_2022_10_02_01_44_19)





4. Future Improvements
    a) Will try to implement n-step bootstrapping
    b) Will try to use array to represent replay buffer’s binary tree
    c) Will try to implement other algorithms like Reinforce and TRPO