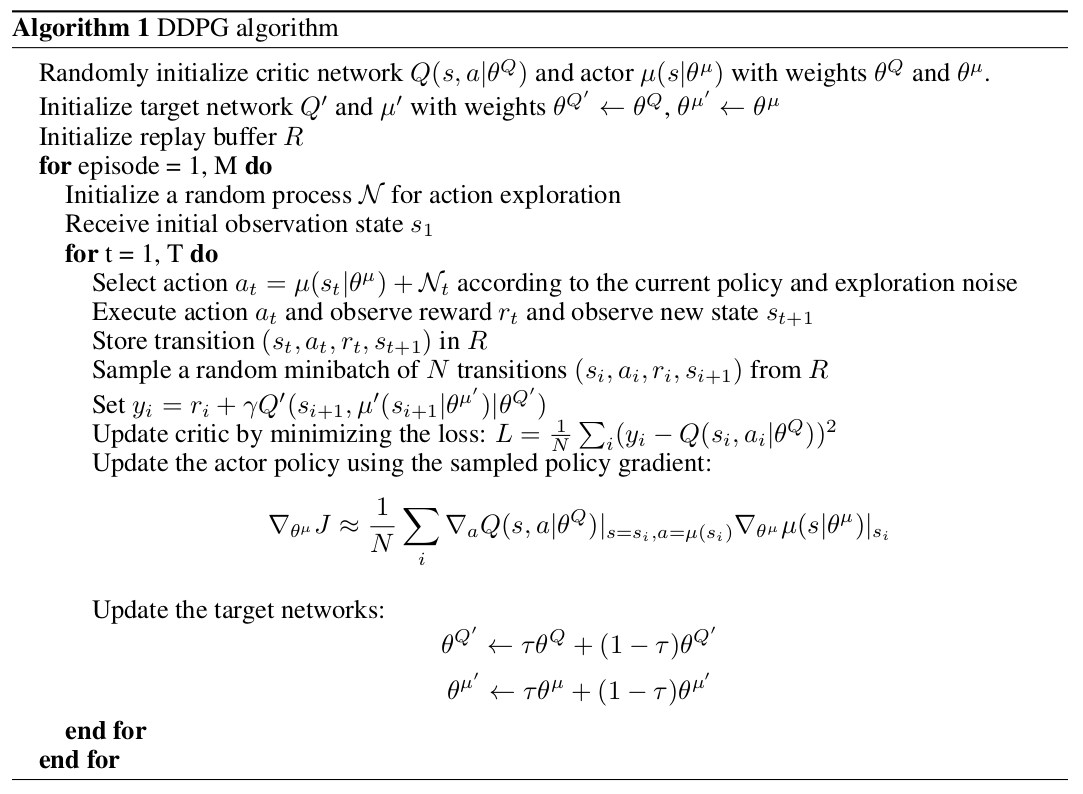
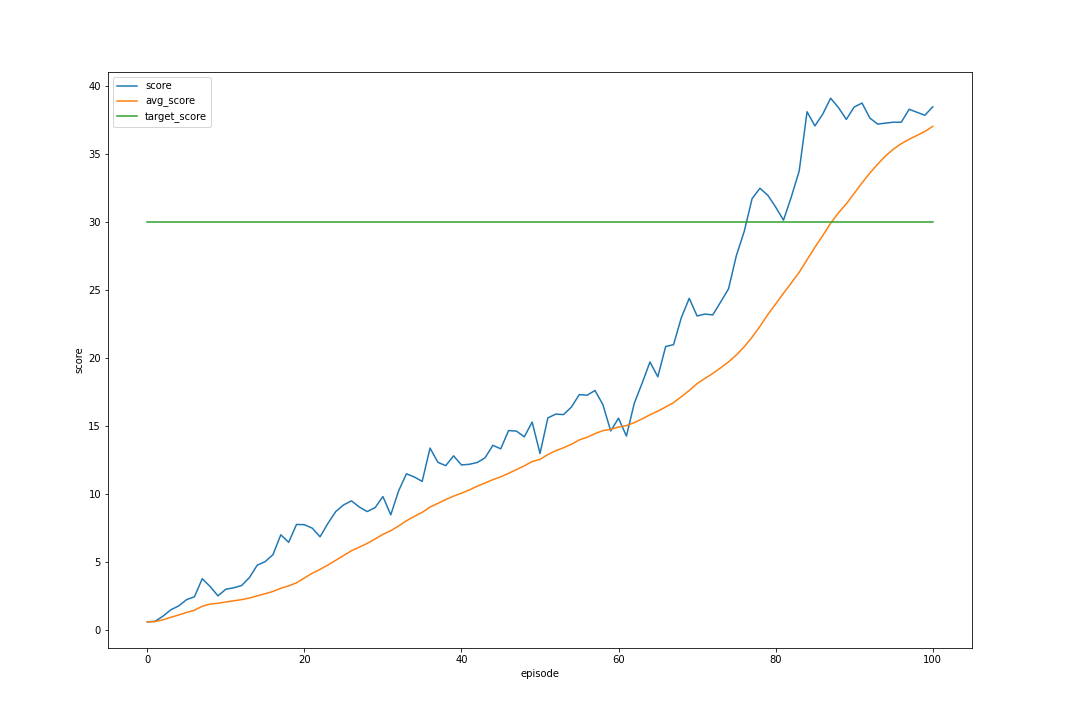
1. Learning Architecture
   1. Algorithm

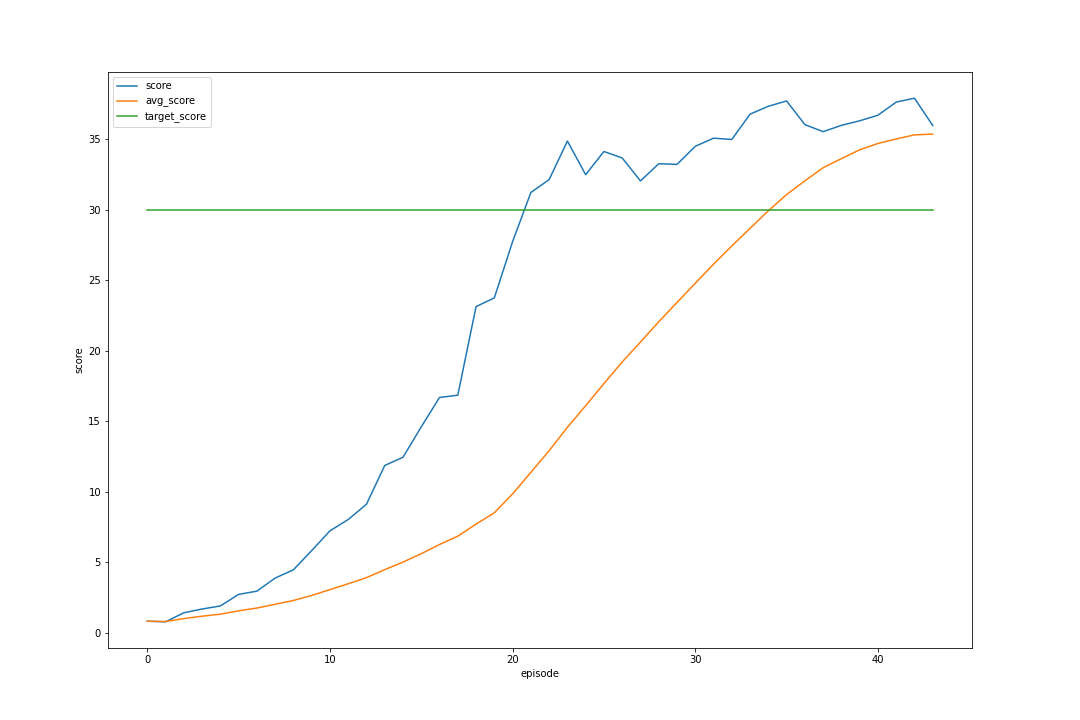


* 1. Model Structure
     1. Actor:
        1. Input size = State size = 33
        2. Hidden layers(2)
           1. Fully connected with 128 batch-normalized rectifiers
           2. Fully connected with 256 rectifiers
        3. Output size = Action size = 4
     2. Critic:
        1. Input 1 size = State size = 33 at 1st layer
        2. Input 2 size = Action size = 4 concat to 2nd layer
        3. Hidden layers(2)
           1. Fully connected with 128 batch-normalized rectifiers
           2. Fully connected with 256+4 rectifiers
        4. Output size = Value function = 1
  2. Hyperparameters
     1. Batch size 128
     2. Memory buffer size 1e6
     3. Number of episodes 1000
     4. Target score 30.0
     5. Discount factor gamma 1e-3
     6. Learning rate for Actor 1e-4
     7. Learning rate for Critic 1e-3
     8. Update Period 20(5 for prioritized exp replay)
     9. Update Times per update 7(1 for prioritized exp replay)
     10. Weight Decay 0
     11. Agent number 20
     12. Alpha(prioritized exp replay) 0.7
     13. Beta(prioritized exp replay) 0.8

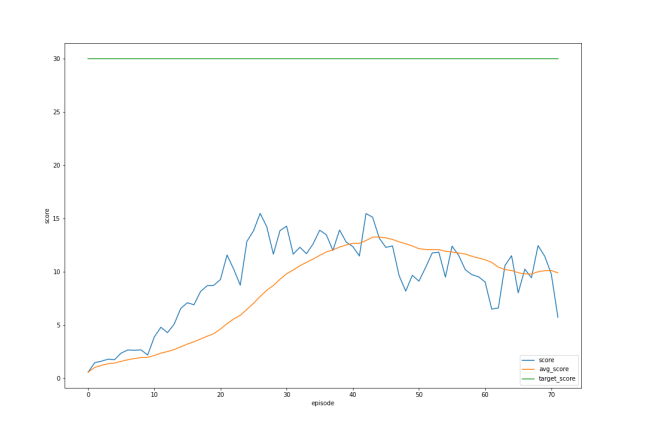
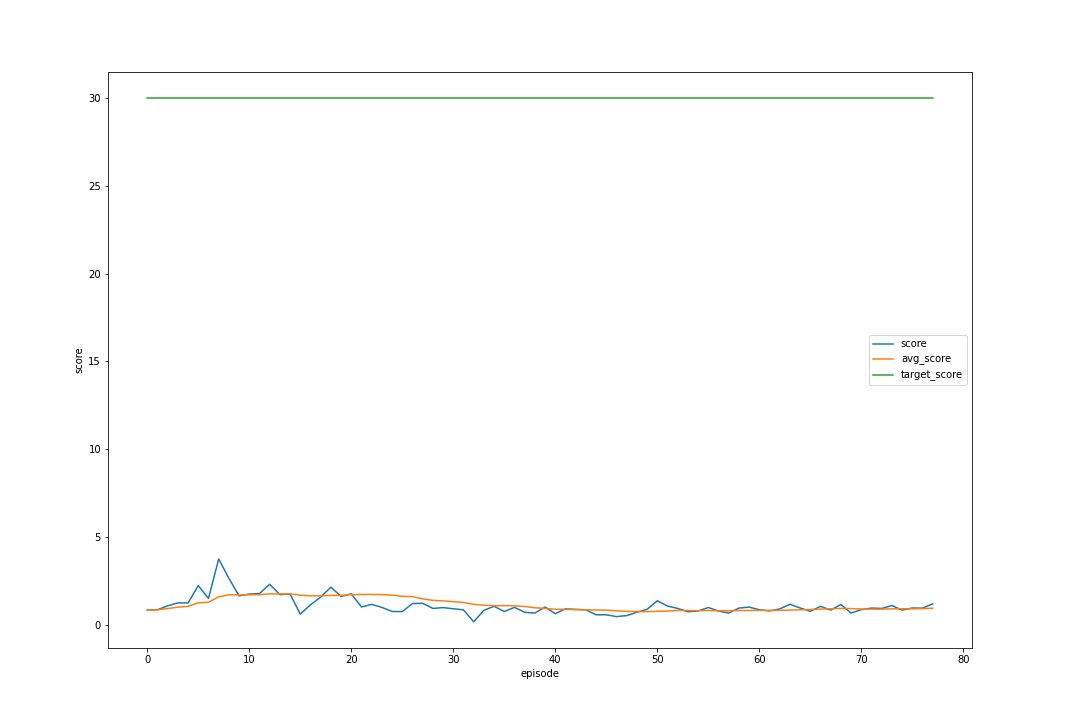
1. Results
   1. Original DDPG
      1. Folder: (ddpg\_result\_2022\_10\_04\_16\_14\_37)
      2. Result:



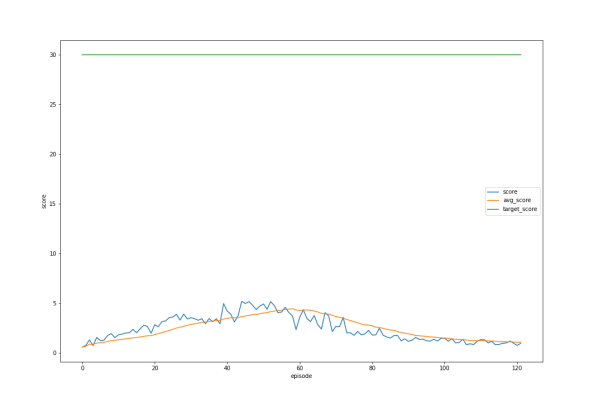
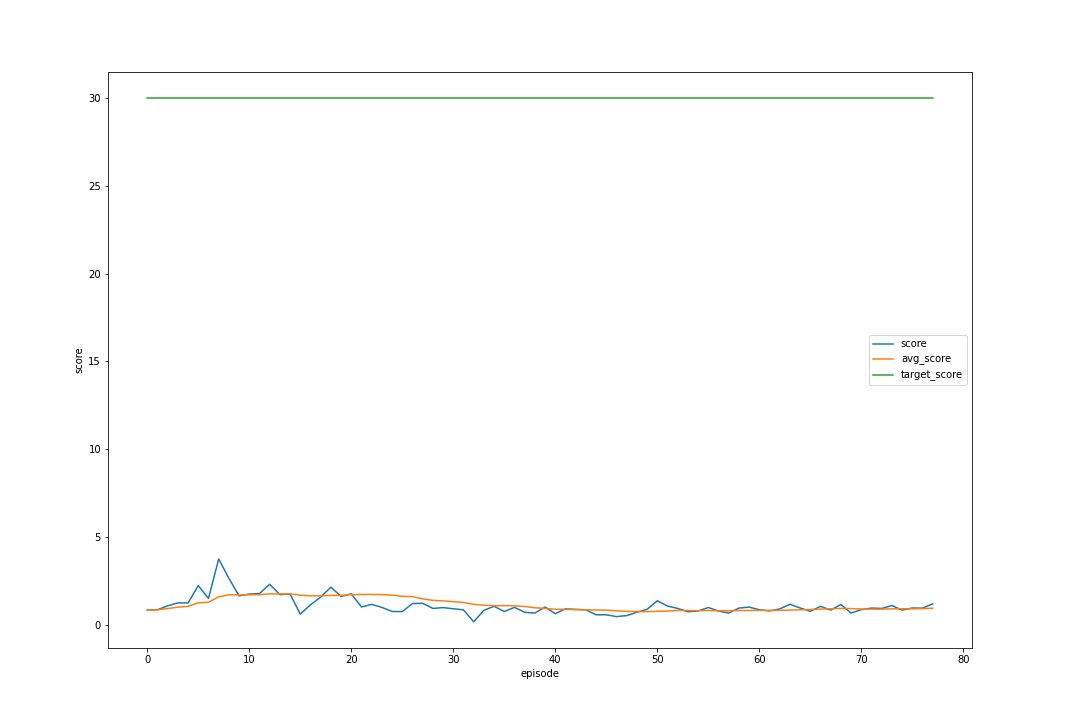
* 1. DDPG with prioritized experience replay
     1. Folder: (ddpg\_result\_2022\_10\_08\_21\_09\_12)
     2. Result:



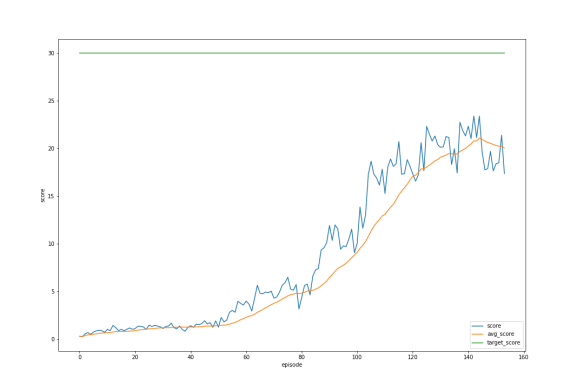
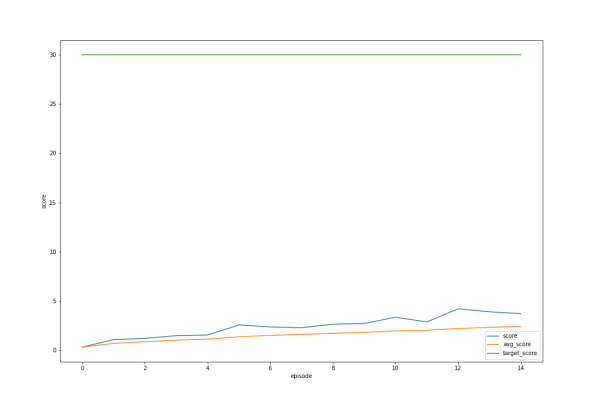
1. Conclusions
   1. Both methods converge within 100 episodes(however in the 2nd one, the training stopped early because a communication error occurred in the unity agent)
   2. Compare to original DDPG, DDPG with prioritized exp replay converge much faster(with in 40 episodes) and require much less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



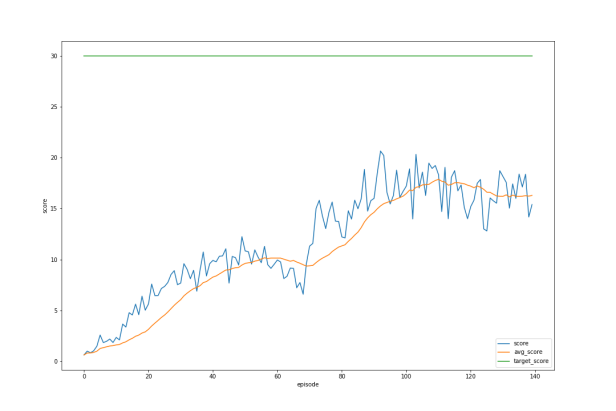
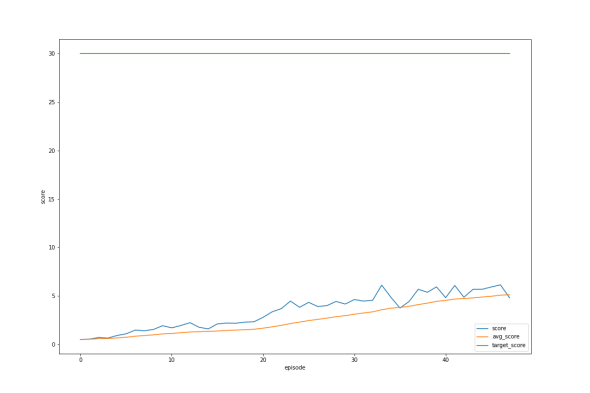
* + 1. Add Batch Normalization
       1. left (ddpg\_result\_2022\_09\_30\_17\_28\_04)
       2. right (ddpg\_result\_2022\_10\_01\_19\_12\_28)



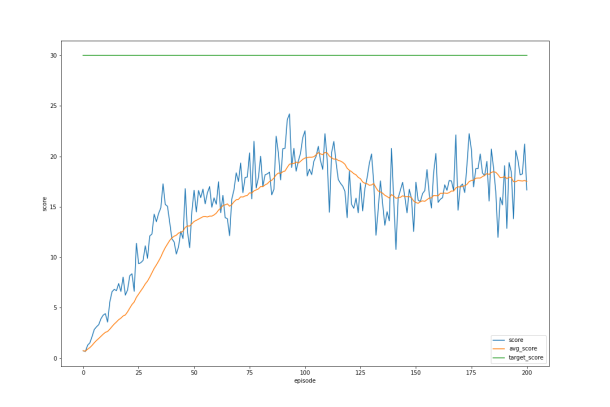
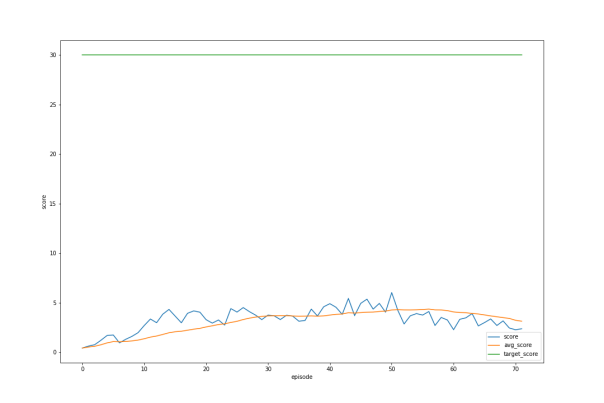
* + 1. Fix OUNoise error by adding -0.5 bias: better
       1. left (ddpg\_result\_2022\_10\_02\_23\_57\_47)
       2. right (ddpg\_result\_2022\_10\_03\_02\_05\_59)



* + 1. 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)



* + 1. 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)



1. Future Improvements
   1. Will try to implement n-step bootstrapping
   2. Will try to use array to represent replay buffer’s binary tree
   3. Will try to implement other algorithms like Reinforce and TRPO