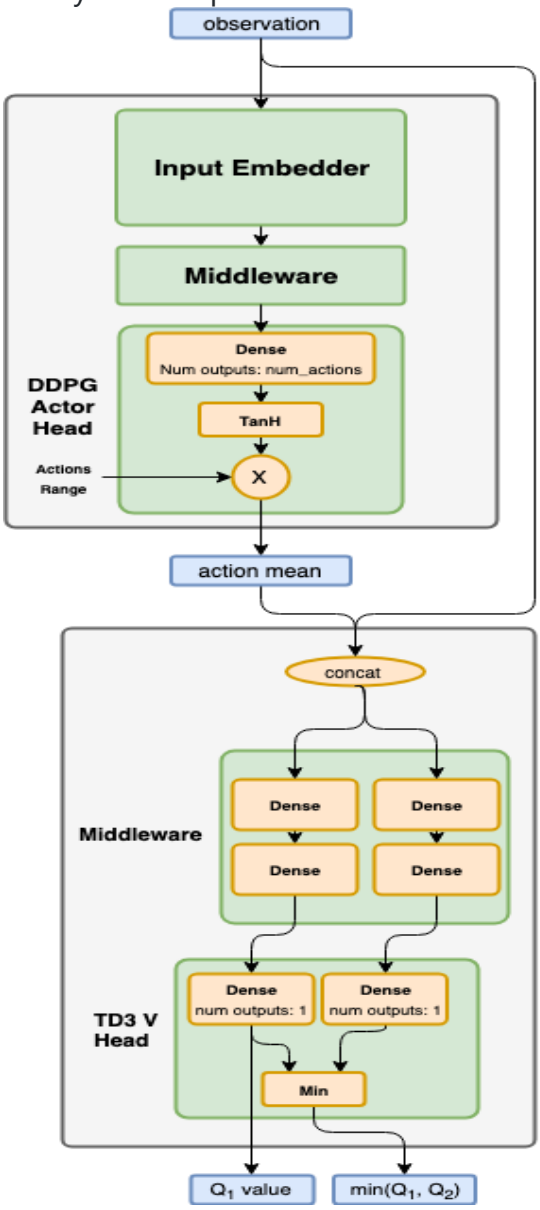
Report of Continues control

1. Define the Network:

- Structure: I use a TD3, Twin Delayed Deep Deterministic policy gradient in this practice [1], since it significantly improves the training performance compare to DDPG: Better convergence in small network and much faster, got 30+ only in 100 episodes.



Graph1, structure of TD3. [2]

- Network size:
  As the Vector Observation space size in this environment is 33, action space size is 4.

  I have defined First hidden layer units as 256 and 2nd hidden layer as 64 for Actor, small enough to run in CPU with the balance of simplicity and efficiency.

  For the Critic First hidden layer units as 128 and 2nd hidden layer as 32, which the similar size as the one in my Navigation project with DDQN. And Critic in TD3 has 2 heads Q1 and Q2, which have identical layers

2. Define the Replay Buffer:

- Make a deque for memorizing episode, size is very large, 1e5 or 1e6

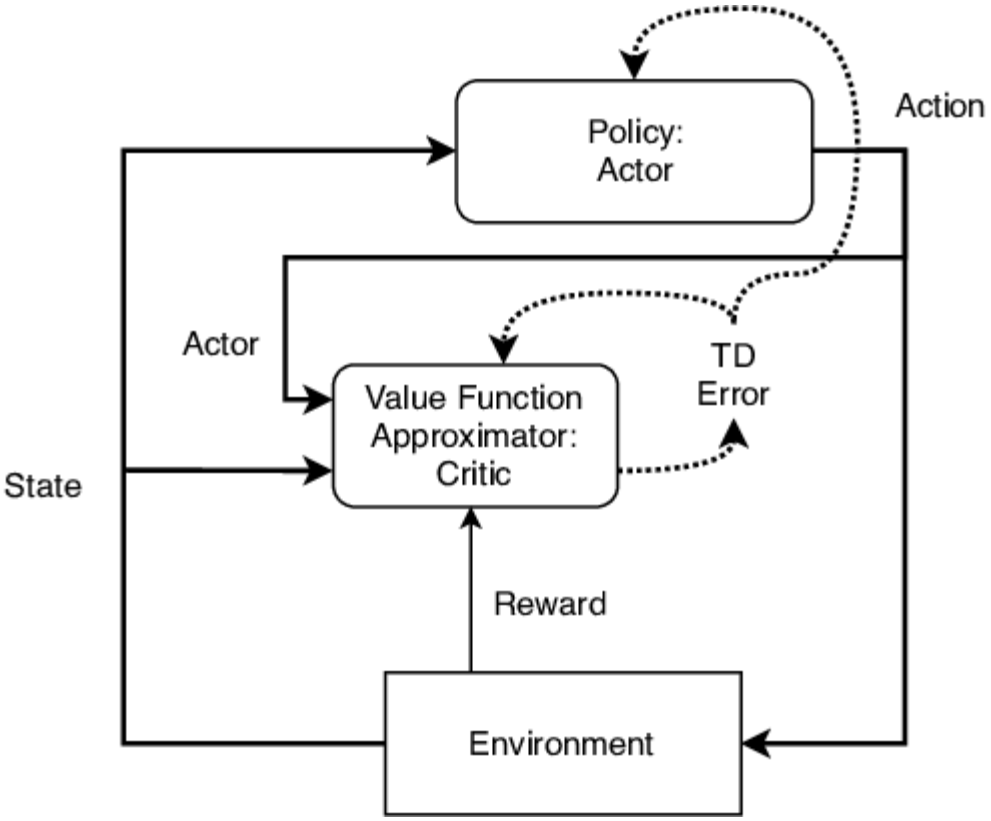
- In each step of the episodes, the information of the 20 agents will be saved to the buffer, includes: State and next State of each agent, action, reward and done.

- Sample randomly from the buffer in each training step, provided it has sufficient items

- In fact 20 Agent provides much more samples, so converge could be quite faster than 1 agent only.

3. Define the Agent:

- Since TD3 is off-policy learning, I need to define 2 set of Networks: local and target, the local network will used for generating actions, and the target network will be used as updating the policy



- TD3 concurrently learns two Q-functions, Q1 and Q2, by mean square Bellman error minimization, in almost the same way that DDPG learns its single Q-function

- In TD3, there is a delaying policy update, which makes TD3 special [1], it will soft update the target network every D steps, D=2 in my case. Soft update is a special way in TD3/DDPG, rather copy the whole local to target, the soft update the new target with

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$
$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$

- Exploration, in TD3, I did not use epsilon-greedy, refer to sfujim's implementation [3], I also use a random in the beginning of each episode, and then add a small noise afterward to maintain the stochastic

- Also, when generate next action with actor target, TD3 will add a noise clip as well,

$$\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$$

4. Training the DDQN:

- Due to the Network is small, and most time was consumed in the simulator, it didn't make much difference in CPU and GPU, sometimes it can reach 30+ less than 100 episodes, and sometimes ~150 episodes.

- Max_Steps: 1001 as observed in the env, it will be finished in 1001

- Skip_timesteps, 50, mean it will random generate action at beginning of each episode, less than 50 would be too small to explore.
  I have tried several numbers, if it's too small, the converge speed would be slow, and if it's too big, it will be hard to converge after 35

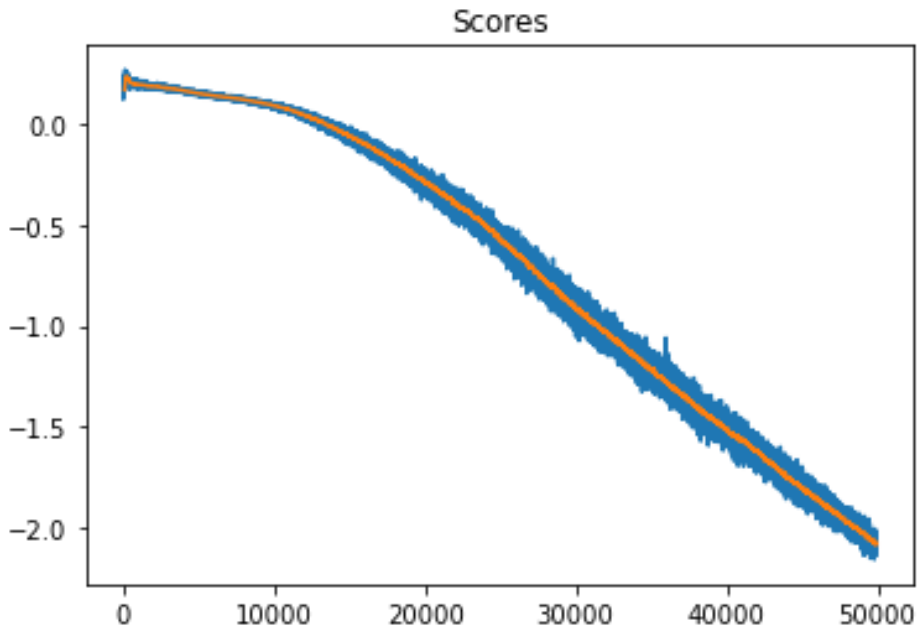
- Random action generation, I made a test before the training, random normal distribution would get more score (0.4+) than random uniform distribution (0.2-).

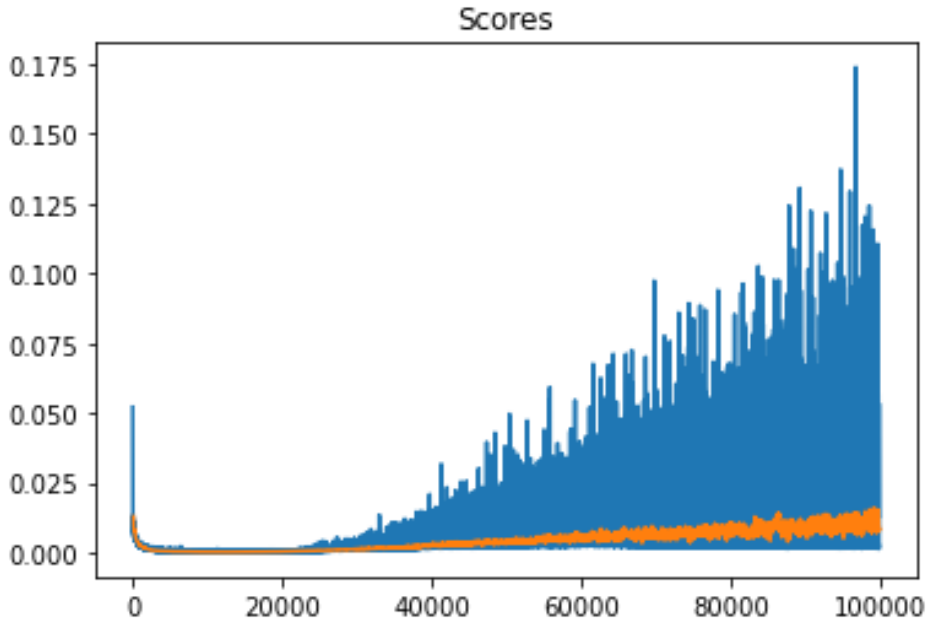
- Batch_size, 128 or 256, 256 is better in convergence.

```
Episode 10      Average Score: 1.10
Episode 20      Average Score: 5.73
Episode 30      Average Score: 17.67
Episode 40      Average Score: 26.48
Episode 50      Average Score: 31.21
Episode 60      Average Score: 32.64
Episode 70      Average Score: 33.25
Episode 80      Average Score: 35.85
Finished at Episode 84  Reach Average Score: 36.03!
```

5. Interesting observations:

- It is very strange that critic loss would keep increasing after a period, while the actor loss keeps reducing, even cross the zero then became negative. I have searched online, seems quite common in DDPG, but mine is creasing so significant, would like to understand more about it.





6. Reference:

1. **"Addressing Function Approximation Error in Actor-Critic Methods"** https://arxiv.org/abs/1802.09477
2. TD3 Network structure, https://nervanasystems.github.io/coach/components/agents/policy_optimization/td3.html
3. Sfujim's TD3 implementation https://github.com/sfujim/TD3/blob/master/TD3.py