Project 2 “Continuous Control”

Dongsu Du

1 Introduction

I decide to use ddpg model for this continuous control project since its simplicity. In the benchmark lecture, the instructor also gave some guideline for ddpg model. What I did is

1. Define the Network architecture in ***model.py***
2. Define the Agent in ***ddpg_agent.py***
3. Train the model in ***Continuous_Control.py***
4. Modify the Network, Agent step function, Network parameter update frequency, for testing the best performance of the ddpg model.
5. Test the model using CPU, once the model is fixed, train the model with GPU.

1 Network Architecture

I tested to use both 3 and 2 hidden layers in this project. I first tried 2 layer network structure with 128/64 and 400/300 hidden layers for both Actor and Critic network. The score stop increase after it reached ~20. Then I tried 3 layer network to see if I can improve the training model. For 3 layers network, I tried 1st hidden layer 128, 2nd hidden layer 64, 3rd hidden layer 32 for both Actor and Critic network.

I tried to reduce the total optimization parameters to make the training time shorter. The activation functions I selected are: Actor: RELU, Critic: RELU.

These choices can give me pretty good results without too much optimization time (or iterations, only 50 epochs). Relu function is selected as the activation function between layers. Please check result session for Network structure testing.

2 DDPG Agent

I selected to train the multi-agents version of the environment. In my agent, the 20 agents add their experiences (20 time steps) into the replay-buffer together (in 1

time step) and the Network parameters are updated 10 times in each 20 time steps. The clipping of the Critic Agent parameter was performed in each step.
To show the importance of clipping, I also test the training without clipping of the Critic Agent parameter.

3 Training process

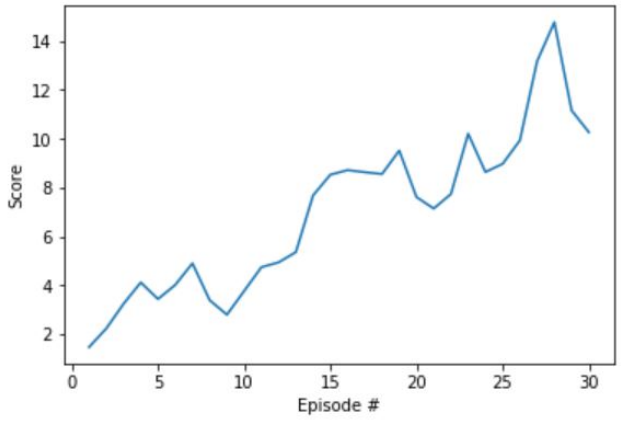
The training process is defined in "Continuous Control.ipynb". The max_t=1000, and the n_episodes is setup to 50. The network will be saved if the mean score value in last 100 episodes is reaching 30.

4 Result

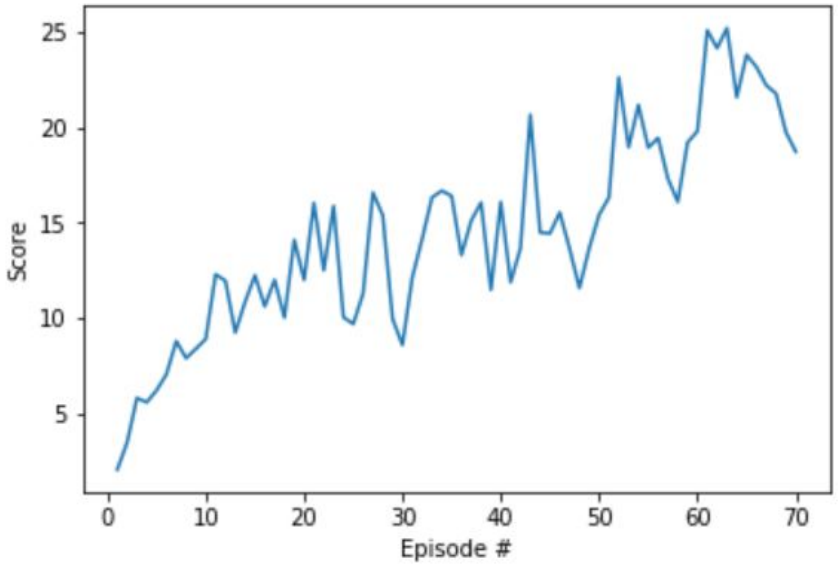
4.1. Training time depends on the frequency of network updates and the network structure. Please see table 1 for training speed on GPU (for 2 layers network). The training time for 3 layer network (128/64/32) is on CPU.

Hidden layers	400/300	128/64	128/64/32
Update frequency	10 per 20 timesteps	10 per 20 timesteps	10 per 20 timesteps
Training time/ Episode	150-200s (GPU) >230S (CPU)	110s-170s	120s-300s (CPU)

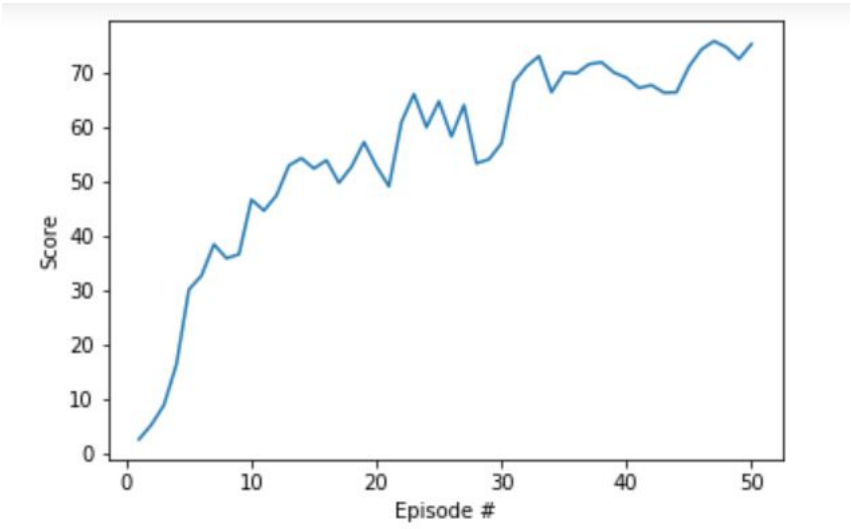
4.2. The score stopped increase at score 14 after 25 episodes during my first training for network (Network: 128/64).



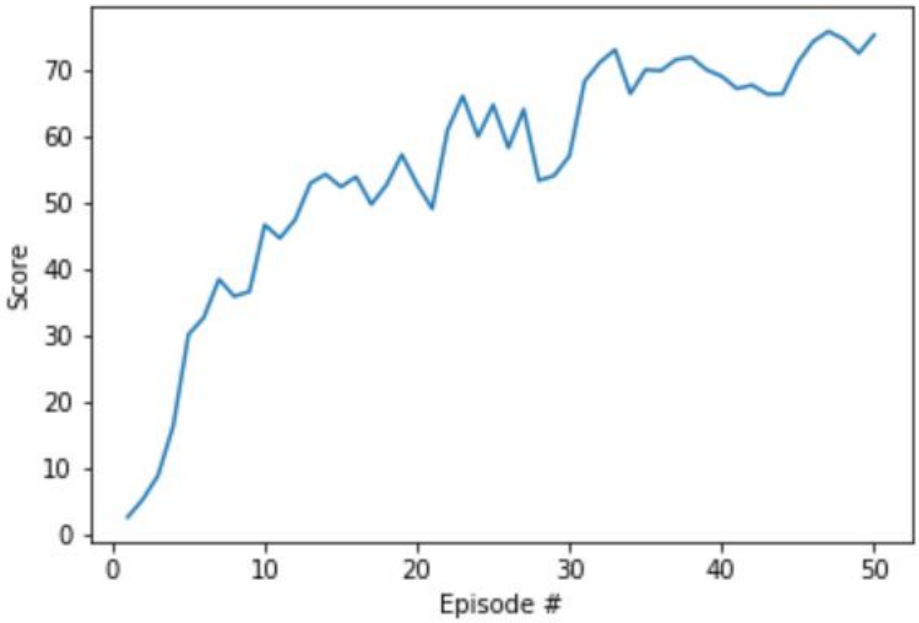
4.3. The score stopped increase at score 25 after 50 episodes during my test training for Network: 400/300



4.4. The average score reached 30 within 50 epochs using network 128/64/32. The max score reaches >70



4.5 Network 128/64/32 without clipping of the Critic parameters.



5 Discussion

The 3 hidden layer shows better performance in model building. And the parameter clipping in 3 layer model is not significant.