CS 533

Tenghuan Li; liten@oregonstate.edu; 933638707

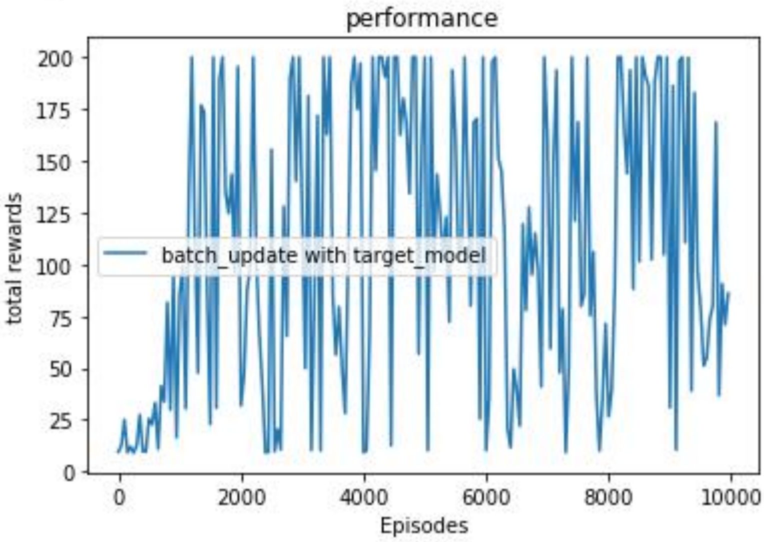
HW4 Distributed DQN

Part1:

1. DQN without a replay buffer and without a target network.

epsilon_decay_steps = 100000, final_epsilon = 0.1, batch_size = 1, update_steps = 1, memory_size = 1, beta = 0.99, model_replace_freq = 2000, learning_rate = 0.0003, use_target_model = False.

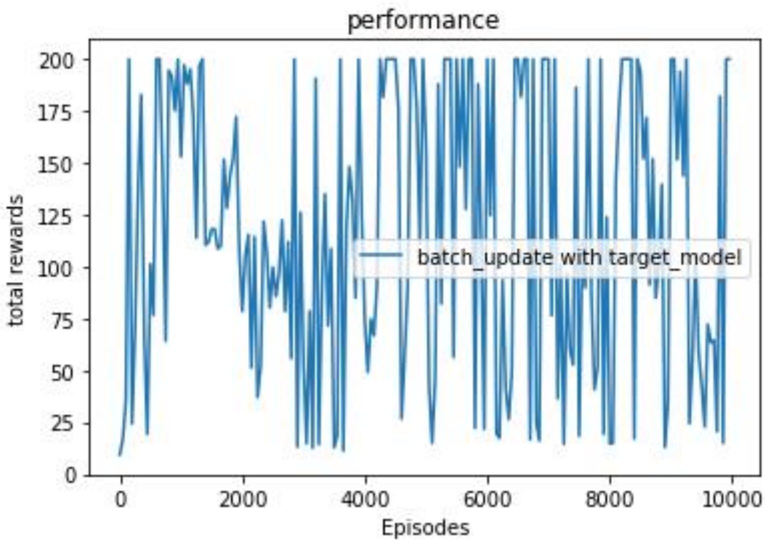
```
Learning Performance:

<Figure size 432x288 with 0 Axes>
```



2. DQN without a replay buffer (but including the target network)

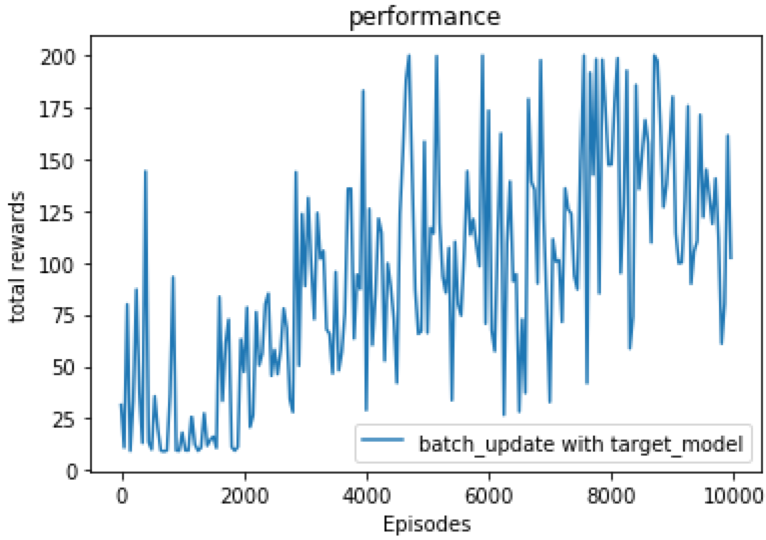
epsilon_decay_steps = 100000, final_epsilon = 0.1, batch_size = 1, update_steps = 1, memory_size = 1, beta = 0.99, model_replace_freq = 2000, learning_rate = 0.0003, use_target_model = True.

```
<Figure size 432x288 with 0 Axes>
```



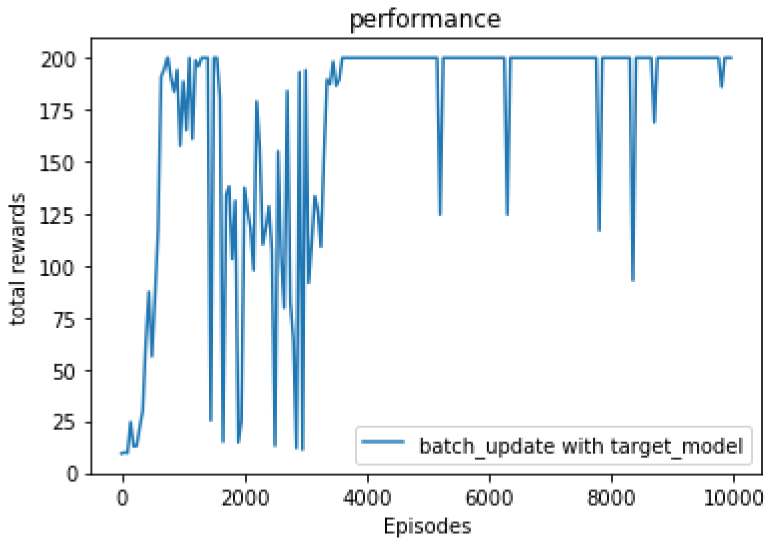
3. DQN with a replay buffer, but without a target network.
epsilon_decay_steps = 100000, final_epsilon = 0.1, batch_size = 32, update_steps = 10, memory_size = 2000, beta = 0.99, model_replace_freq = 2000, learning_rate = 0.0003, use_target_model = False.

```
<Figure size 432x288 with 0 Axes>
```



4. Full DQN
epsilon_decay_steps = 100000, final_epsilon = 0.1, batch_size = 32, update_steps = 10, memory_size = 2000, beta = 0.99, model_replace_freq = 2000, learning_rate = 0.0003, use_target_model = True.

```
Learning Performance:

<Figure size 432x288 with 0 Axes>
```



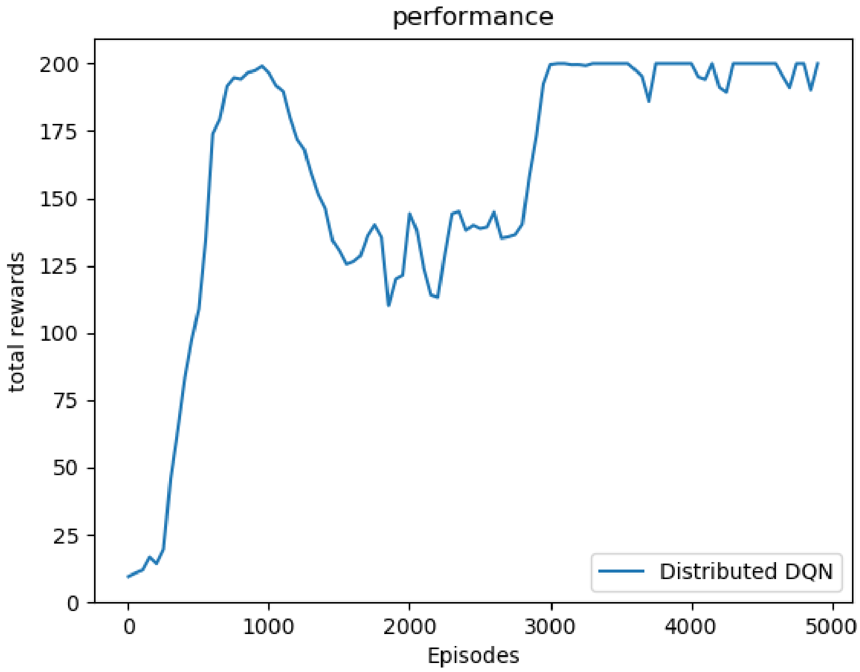
Summary:

To compare these four pictures, we can see that the first one DQN without replay buffer and without target network This graph has the worst average reward, and you can see that the learning is the worst. The second picture the performance is better than the first without replay buffer, but the overall is still poor. Comparing the 1 and 3 we can see that with replay buffer the rewards is increase progressively. Comparing the 2,3 and 4, we can find that full DQN which have replay buffer and target network it gets the best performance. Then, we can know for DQN, it needs the replay buffer and target network both work together to get better performance and the appropriate learning curve.

Part 2: Distributed DQN

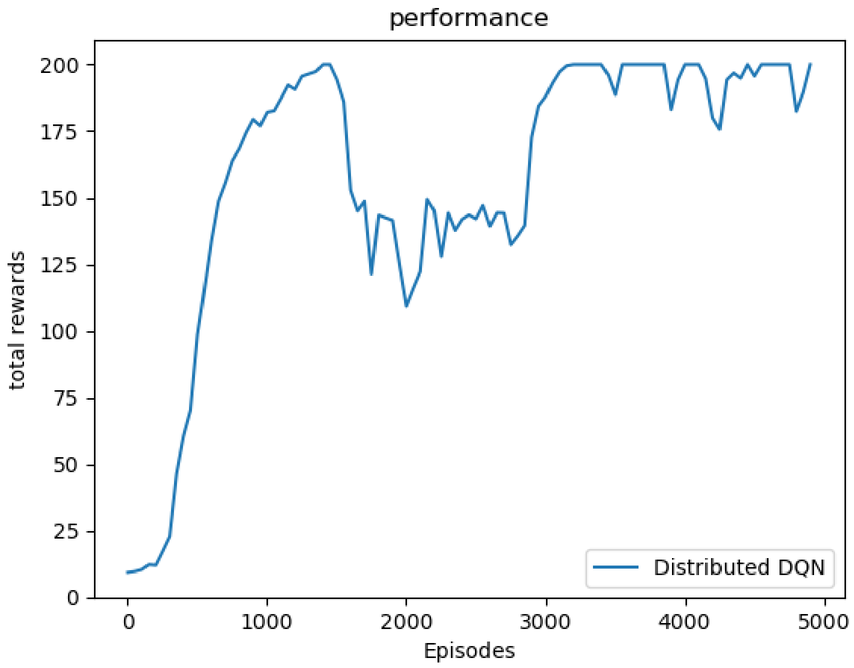
1. When the collector worker = 4, evaluator worker = 4, training_episodes = 5000, test_interval = 50.

The total running time is 2659s.



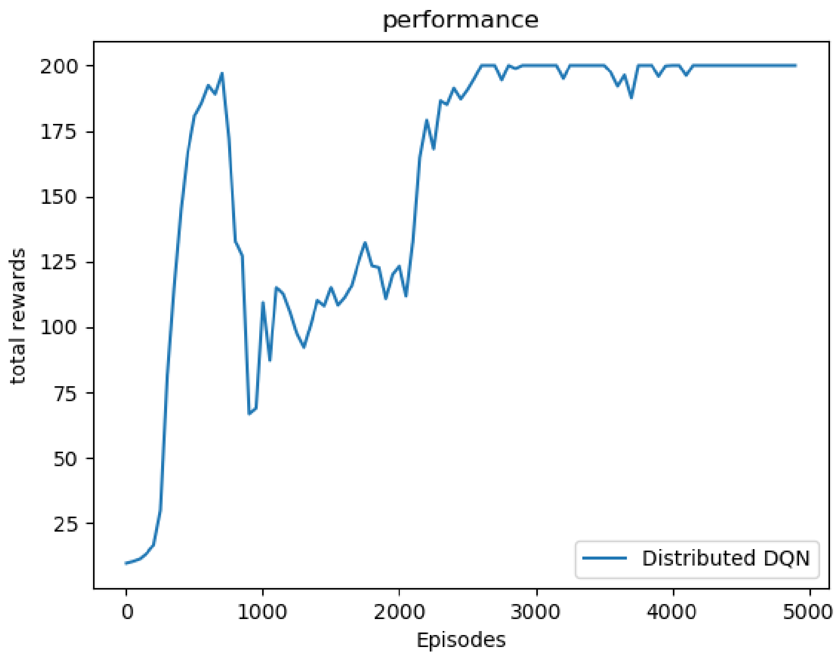
2. When the collector worker = 4, evaluator worker = 8, training_episodes = 5000, test_interval = 50.

The total running time is 2545s.



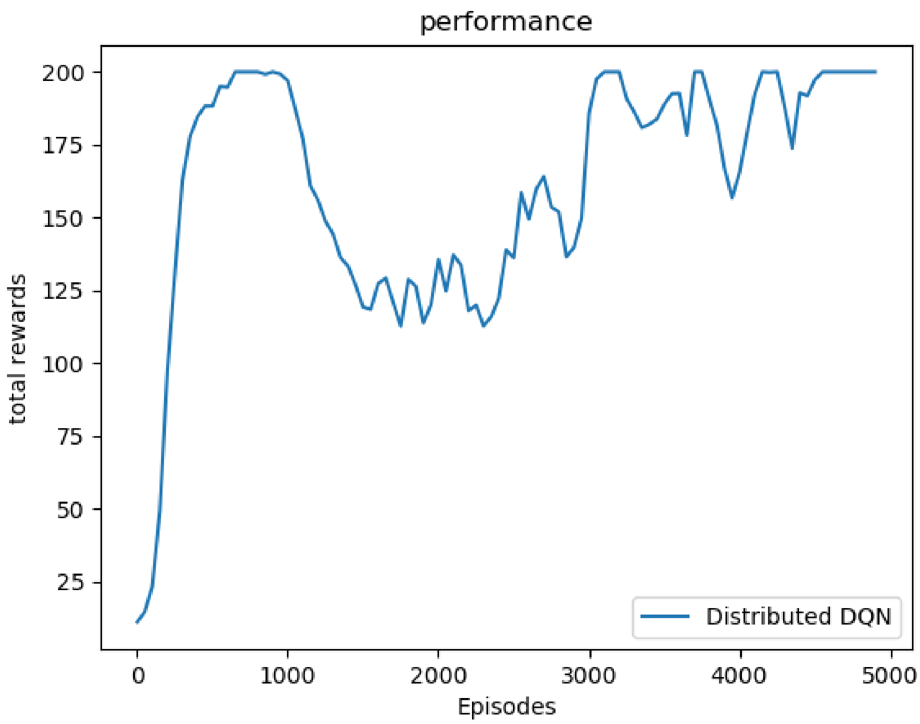
3. When the collector worker = 8, evaluator worker = 4, training_episodes = 5000, test_interval = 50.

The toral running time is 1944.23



4. When the collector worker = 8, evaluator worker = 8, training_episodes = 5000, test_interval = 50.

The total running time is 1538s.



Summary:

For my part 2 the distribute DQN, I had run four sets of data which were collector worker, evaluator worker: (4, 4), (4, 8), (8, 4), (8, 8). As the graph and running time shows, we can get that when the collector = 4, the evaluator worker changes have little effect on the running time. As the number of evaluators increases, the computation speed will increase slightly. When the evaluator =4, the collector worker 8 is faster than 4. So, we can get that the more collector worker can make the training speed faster. This fits the concept we learned because the number of collecting workers is running in parallel, and the more workers, the faster the speed. So, the (8, 8) is the faster one in these four sets.