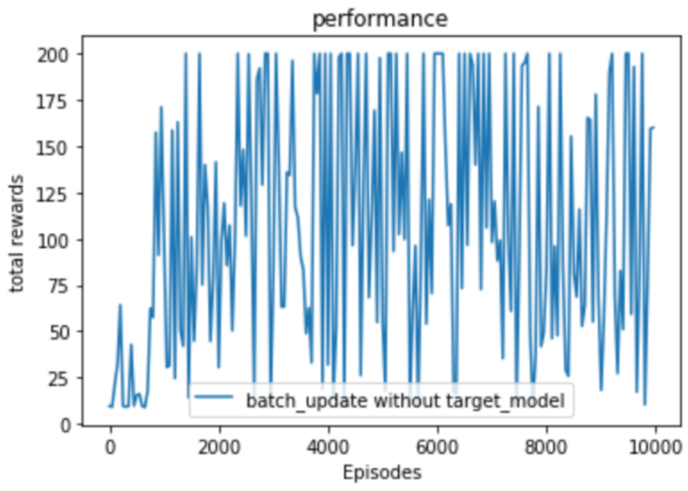
HW4: Deep Q-Learning Network

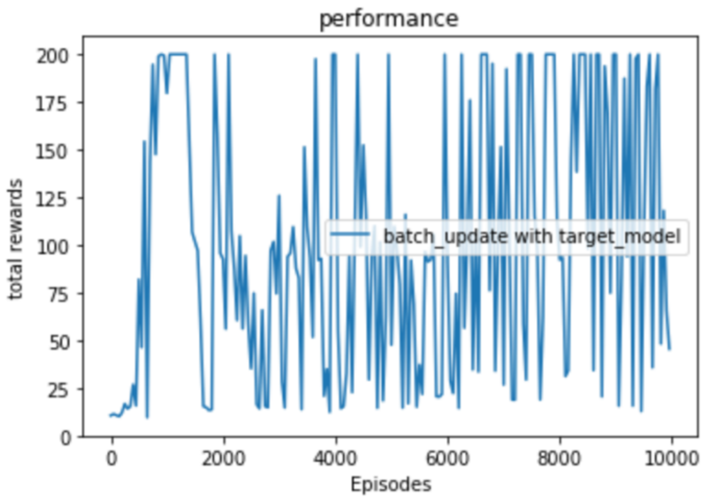
Chih-Hsiang Wang
Chieh-Chin Tao

Part 1. Non-distributed DQN

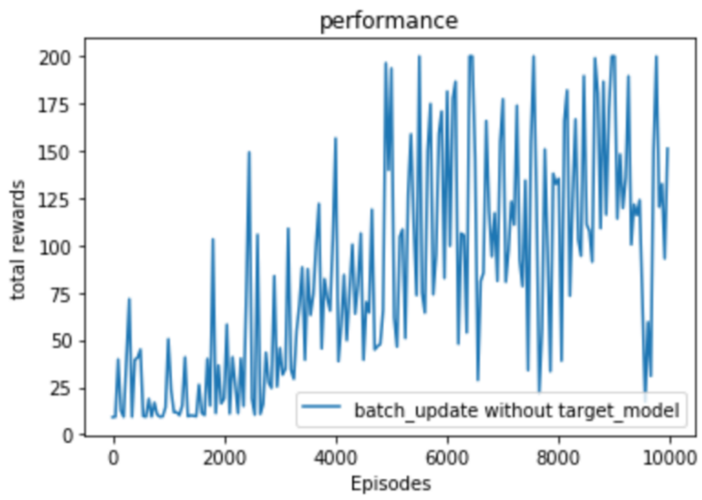
1. DQN without a replay buffer and without a target network.
   (epsilon_decay_steps = 100000, final_epsilon = 0.1, beta = 0.99,
   model_replace_freq = 2000, learning_rate = 0.0003,
   memory_size = 1, update_steps = 1, batch_size = 1, use_target_model = False)



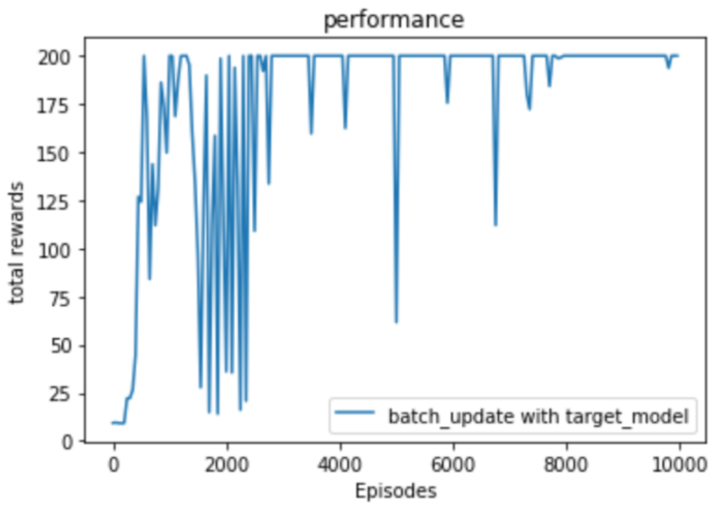
2. DQN without a replay buffer but including the target network.
   (epsilon_decay_steps = 100000, final_epsilon = 0.1, beta = 0.99,
   model_replace_freq = 2000, learning_rate = 0.0003,
   memory_size = 1, update_steps = 1, batch_size = 1, use_target_model = True)



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



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



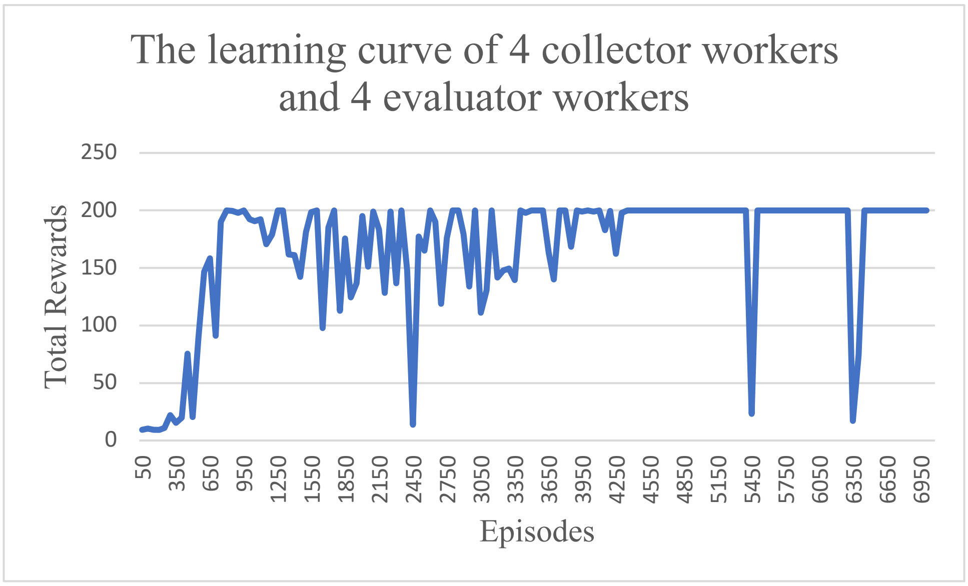
A summary of your observations regarding the differences you observed:

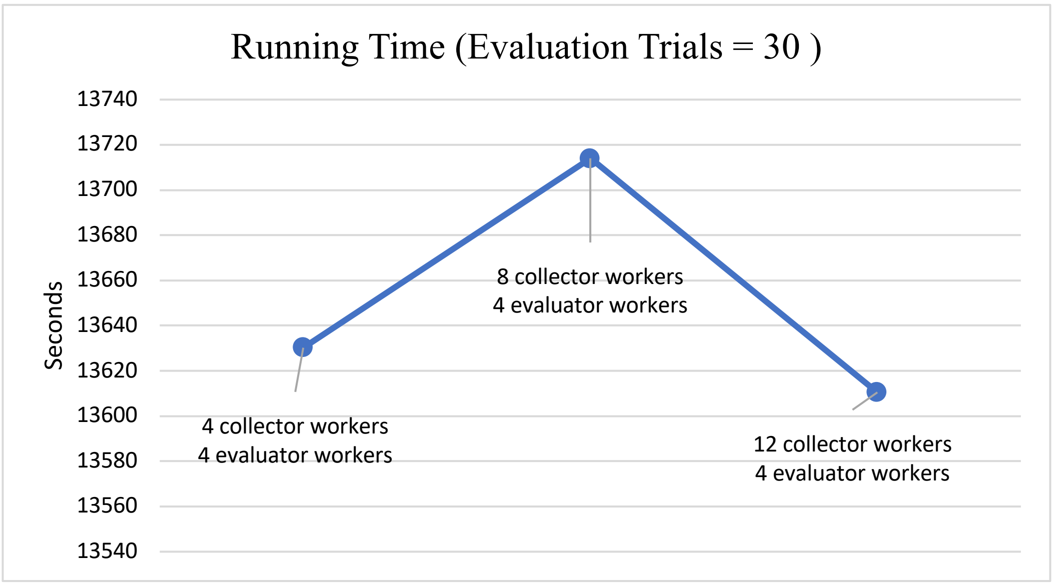
Comparing the running time with these four DQN implementations, the DQN without a replay buffer will take much longer than the DQN with a replay buffer (almost one and half hours longer). Besides, the performance of the DQN with a replay buffer is better than the DQN without a replay buffer. According to the graph, the rewards with a replay buffer increase progressively, but the DQN without a replay buffer doesn't progress. The full DQN which with both a replay buffer and a target network has the best performance, it tends to get maximum reward after running enough episodes. However, the DQN which only has a target network doesn't perform well. It needs both a replay buffer and a target network to achieve an appropriate learning curve.

Part 2. Distributed DQN

In this part, we use two methods because there are tradeoffs between them. The first method is the normal one using exactly the same process as non-distributed deep Q-learning. The learning curve is similar to non-distributed model, but the learning time is slow. The second method lets the collector workers update the batch separately. It will reduce the update number of actions. The learning time for collector workers is much faster, but the learning curve is a little bit different from the non-distributed model.

- Method 1 – normal version

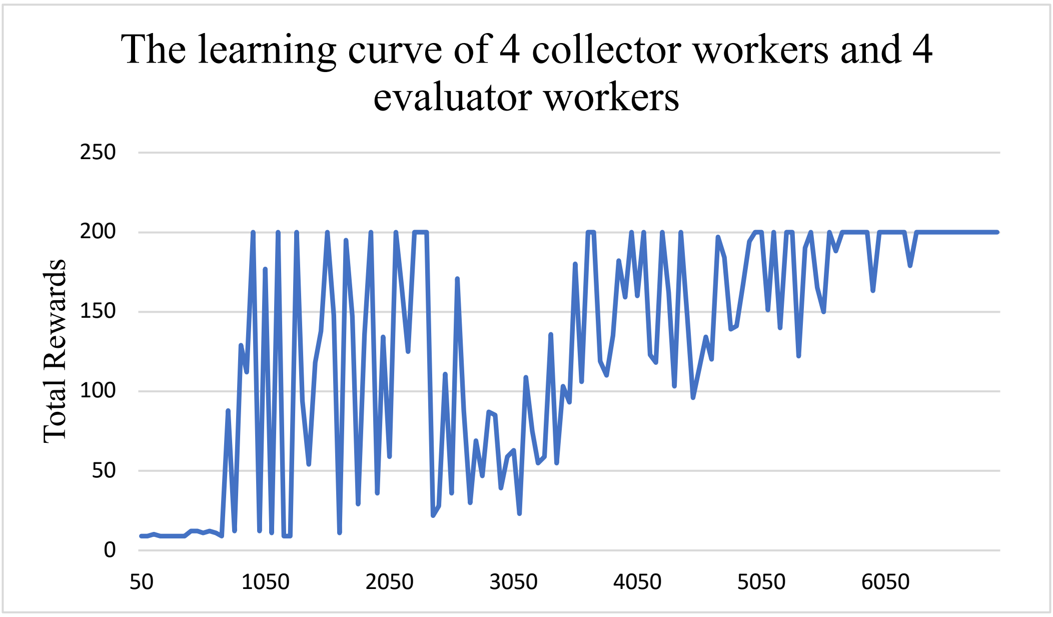


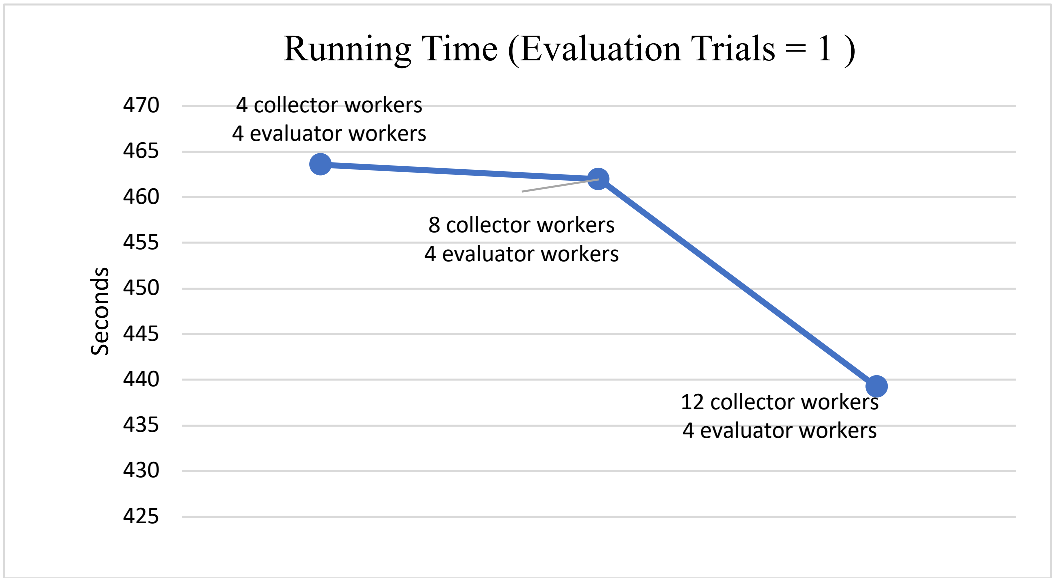


Observations:

When evaluation trial is 30, the running time of 12 collector workers is the fastest, following by 4 collectors and 8 collectors. It may be related to the long evaluation time compared to collecting experience. The program spends much time on evaluation which makes altering number of collector workers less influential. Besides, the collector workers may waste time on communicating with the model server or waiting in the queue causing the increased workers not efficient enough as our anticipation.

- Method 2 – simplified version





Observations:

By using simplified method, due to the fact that the running time of collector workers is much faster than evaluator workers, we change the trials of evaluator from 30 to 1 in order to compare the running time of different number of collectors. From the running time graph of three different sets of workers, we observe that the more collector workers there are, the faster the training will be. It is reasonable because each collector will do their tasks in parallel. However, the communication between workers cause some time wasted, which results in the similar running time between different sets of workers.