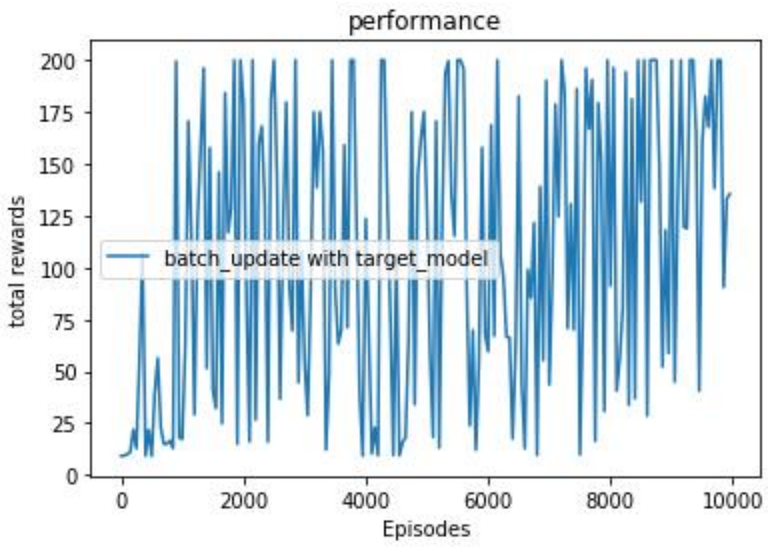
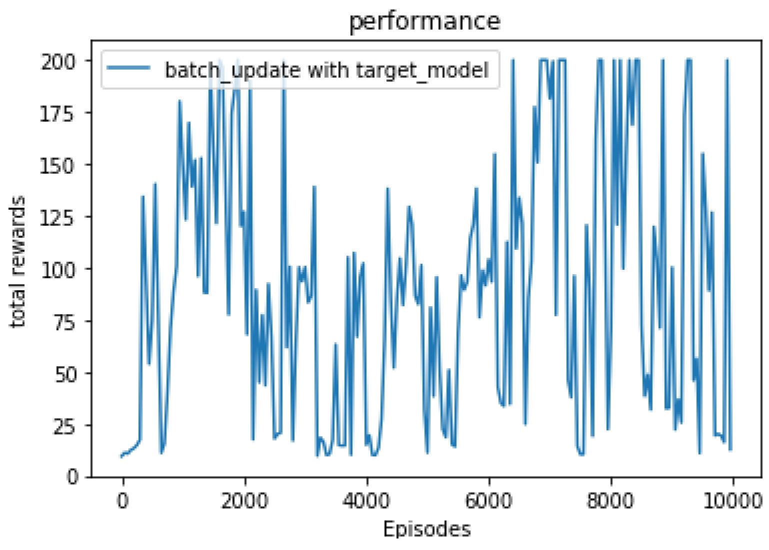
Part 1: Non-distributed DQN

You should conduct the following experiments involving different features of DQN.

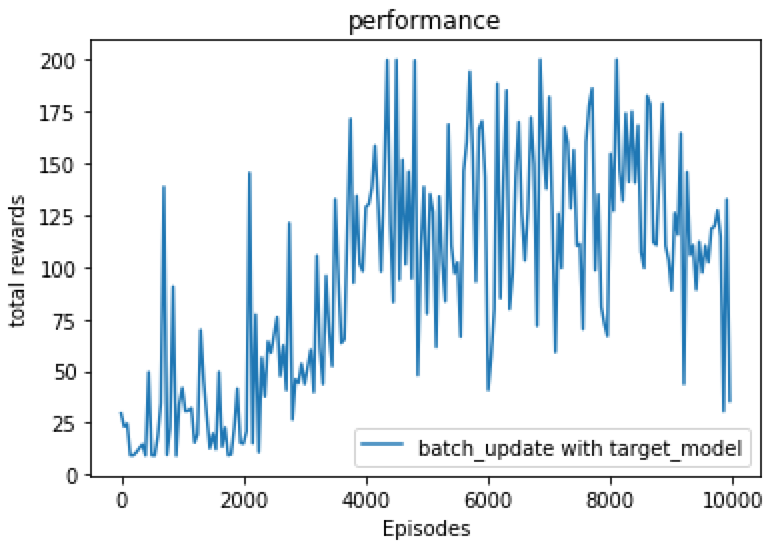
1. DQN without a replay buffer and without a target network. This is just standard Q-learning with a function approximator. The corresponding parameters are: memory_size = 1, update_steps = 1, batch_size = 1, use_target_model = False



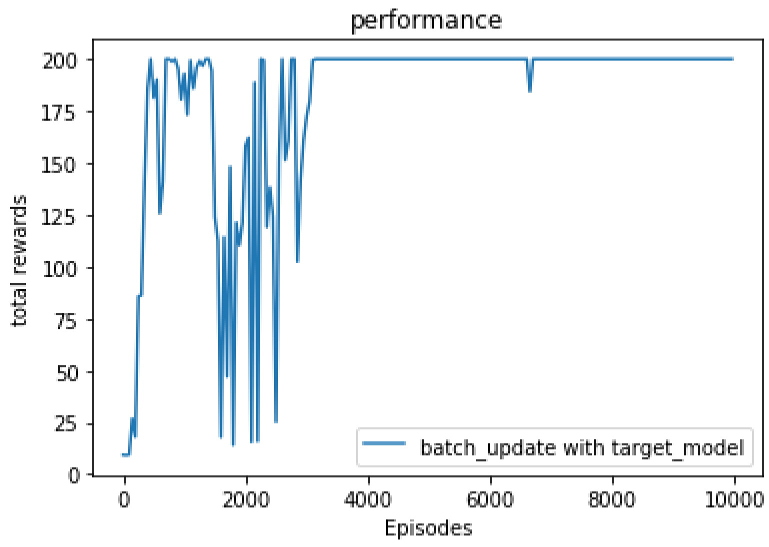
2. DQN without a replay buffer (but including the target network). The corresponding parameters are: memory_size = 1, update_steps = 1, batch_size = 1, use_target_model = True



3. DQN with a replay buffer, but without a target network.
   Here you set use_target_model = False and otherwise set the replay memory parameters to the above suggested values



4. Full DQN



For each experiment, record the parameters that you used, plot the resulting learning curves, and give a summary of your observations regarding the differences you observed.

A. **Parameters:** Based on provided parameters, specific parameters were modified by following the direction of each experiment.

B. A Summary of Observations

1. **DQN without a replay buffer and without a target network:** It took the longest time of four experiments because it does not have the record of training and the target model. Thus, it has to experience each step one by one as you can see in the graph.

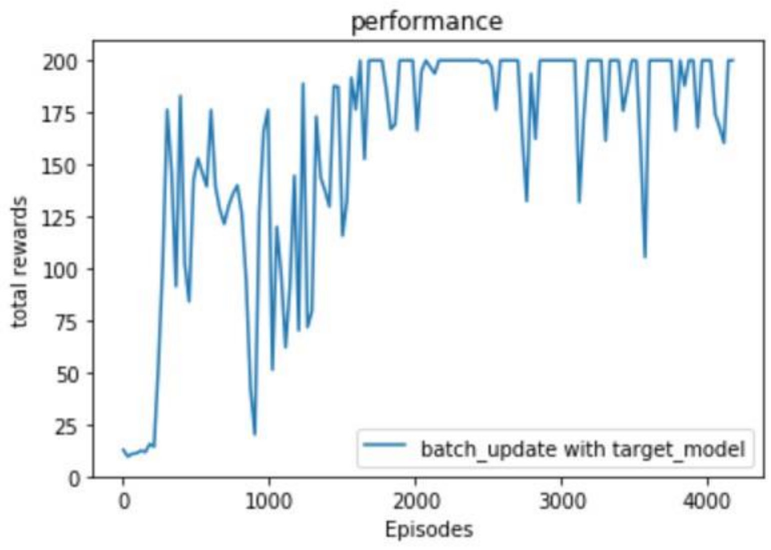
2. **DQN without a replay buffer (but including the target network):** It became little bit faster than the first experiment because storing all of those data in memory is quite costly. However, it needs to go through an extra iteration of prediction.

3. **DQN with a replay buffer, but without a target network:** It still could not have the effective training result.

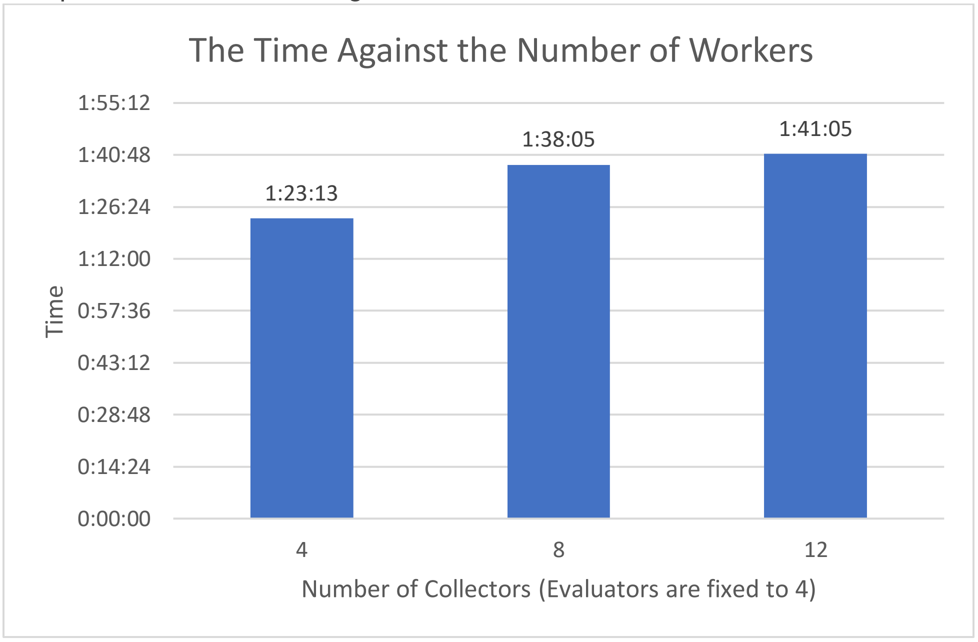
4. **Full DQN:** Adding to the replay buffer, to have the target network multiply its outputs by a "mask" corresponding to the one-hot encoded action is a better strategy as the result. It will set all its outputs to 0 except the one for the action we actually saw. Then, it can pass 0 as the target to for all unknown actions and its neural network should thus perform fine. In order to predict for all actions, it can simply pass a mask of all 1s.

Part 2: Distributed DQN

1. Learning Curve with 4 collectors and 4 evaluators



2. The plot of wall clock time against the number of workers



3. Discussion
   The result of the time depends on the environment of the server in Intel. However, one thing that we can say is that the communication between DQN server and evaluators takes most of time to get the result. Thus, if we can make some improvement in that part in a parallel computing way, we could expect a better result in the total time consumption.