Zehao Li

CID: 01884407

Reinforcement Learning (Second Half) Coursework

Question 1 (i) Apparently, variance of the mini-batch learning is much smaller than that of online learning. And the mini-batch learning leads to a more stable result because the loss fluctuate much less at the end of training. This is because in online learning untrained parts of the Q-network will also be "overwritten" due to that subsequent training samples are highly correlated to the online learning results. Then training on those affected parts can induce large and unsteady changes to the Q-network. While in batch learning the Q-network can be updated entirely, because the batch sampling has a more uniform data distribution, giving steady results.

Question 1 (ii) From both plots (Figure 1 (a) & (b)) it can be observed that both curves have descending trends but the loss of mini-batch learning decreases faster and is much smaller than that of online learning at each episode, which means mini-batch learning is more efficient. The reason may be that the mini-batch learning has a replay buffer containing all experienced transitions, and these transitions are possible to be easily reused for updating the Q-network. However, with online learning, each trained transition is discarded, so the agent has to revisit a state-action pair repeatedly to train on the same transition, which is very inefficient.

Question 2 (i) In this Q-value graph, the bottom-right region leads the agent to go upwards, while the up-right region results in actions of going right. Bottom-right region is likely to show more accurate results than the other. Because the agent starts from bottom-left part, it visits the bottom-right region more frequently when randomly exploring the environment. Therefore, bottom-right Q-values are better learned. There might be only a few episodes the agent could reach the up-right part within the limited episode length, so the Q-values of that region are not well updated.

Question 2 (ii) The agent is not likely to reach the goal under current situation. This is because the Bellman equation has not been applied, and the agent only cares the immediate reward (discount factor=0). Hence, the agent will follow the shortest path to the goal, however ignoring the obstacle. This causes the result that the agent ends in constantly hitting the obstacle instead of bypassing it. (Figure 2(b) shows the fully greedy policy, i.e. $\varepsilon = 0$)

Question 3 (i) The agent needs to take the discount factor and the Q-values of the subsequent state into account with Bellman equation. So when updating the Q-network, the agent is now able to learn from the immediate reward as well as the future return. Without the Bellman equation, however, the future impact of actions cannot be learned for updating the Q-network.

Question 3 (ii) The loss curve with the target Q-network has a more regular pattern, which is that a pulse of curve appears every 10 episodes. This might be the consequence of updating the target network regularly. Every time when the target network is updated, the calculation of loss will be based on its new parameters, which can lead to a sudden change on the loss curve, i.e. the curve pulse. The fluctuation of the loss curve without the target network shows no obvious pattern. And the target network can help to deal with the negative effect of generalization, which makes the loss curve seem more stable at the end than that without target network.

Question 4 (i) If ε is 0, it is impossible that the agent can reach the goal. Because in this case, the actions are chosen fully greedily for each step in episodes, which means the agent cannot explore the environment sufficiently. Therefore, the Q-values of most states have never been learned and updates. The Q-network is inaccurate and the agent will not find the path to the goal based on this.

Question 4 (ii) Although the agent reaches the goal, that episode may still continue. The goal state also has Q-values, and in this case the agent is led to the left by those values. Since the Q-network has not yet converged, the goal state's left region may not be well learned. So Q-values of that region may not be accurate, causing the agent to go in straight and hit walls. But when the network has converged, the agent is likely to repeatedly move between the goal state and its adjacent state.

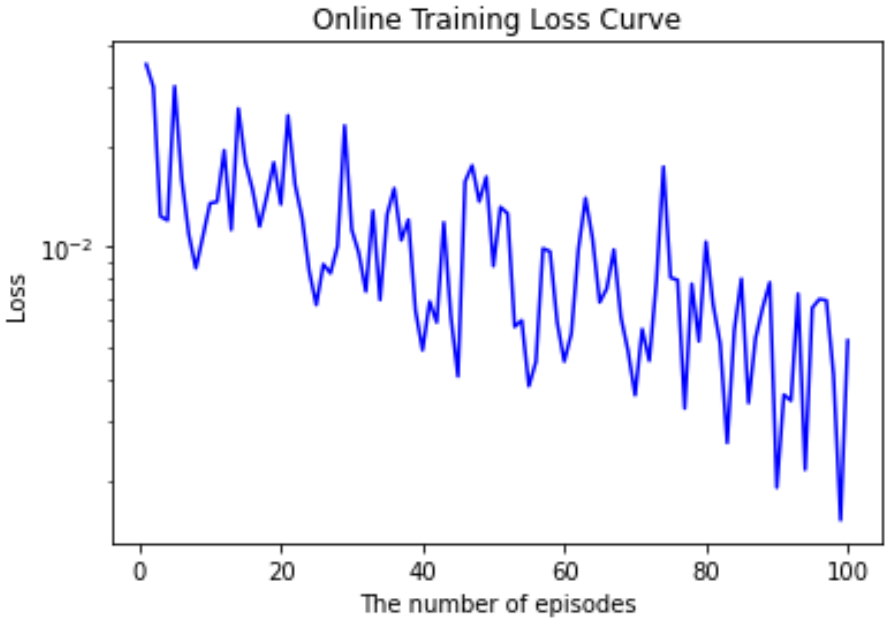

Figure 1 (a)

Figure 1 (b)

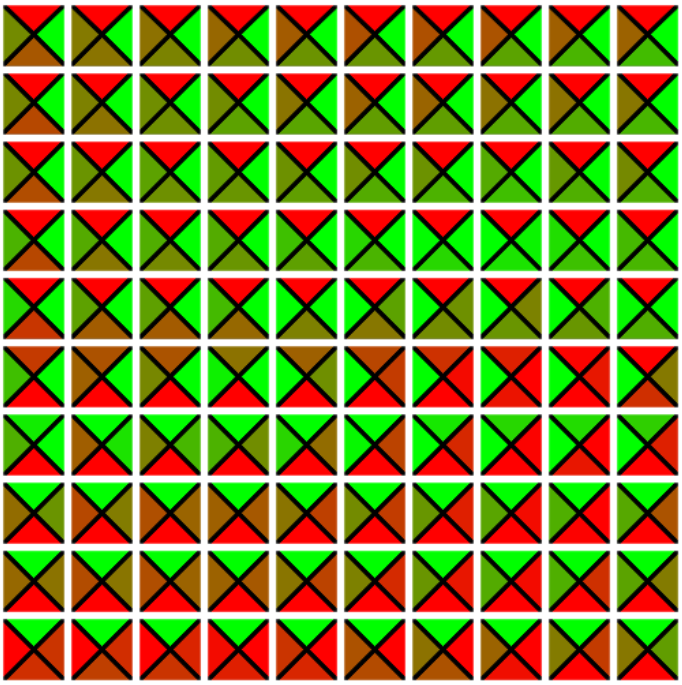

Figure 2 (a)

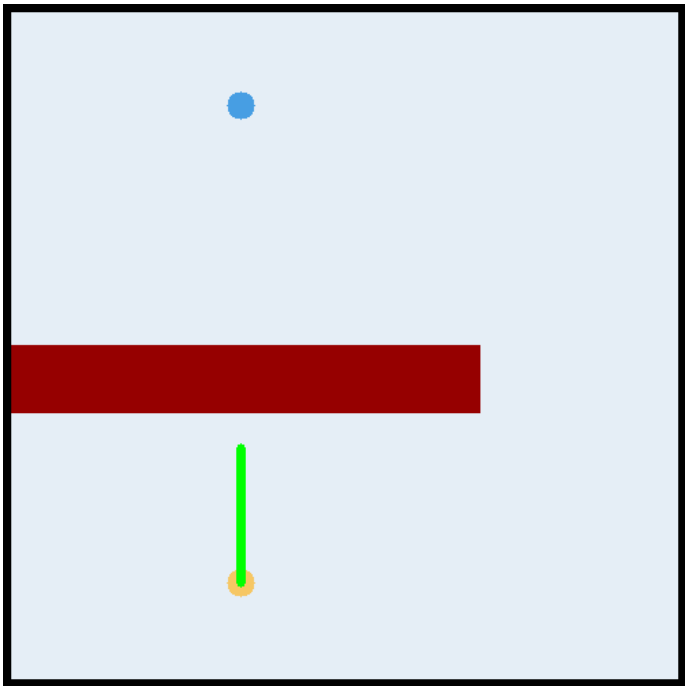

Figure 2 (b)

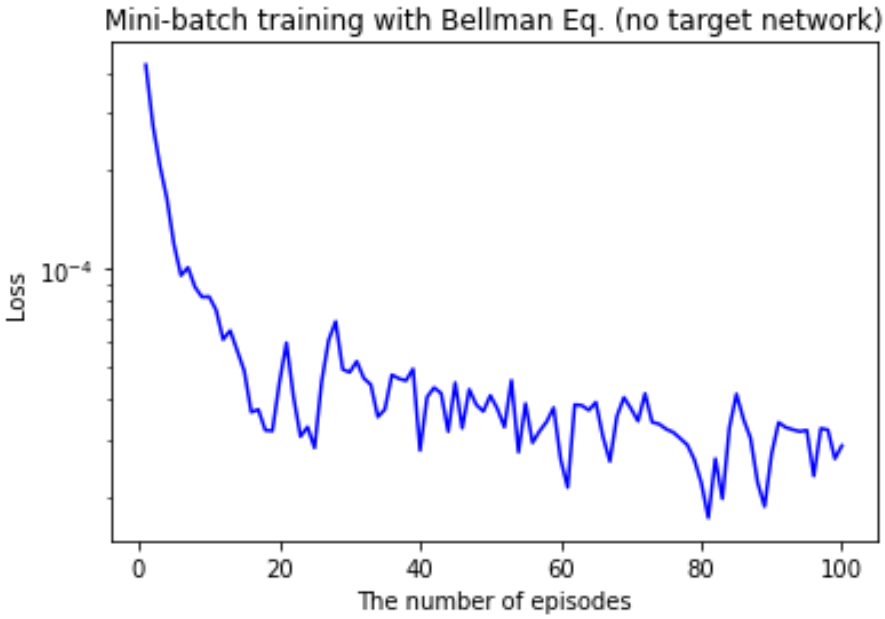

Figure 3 (a)

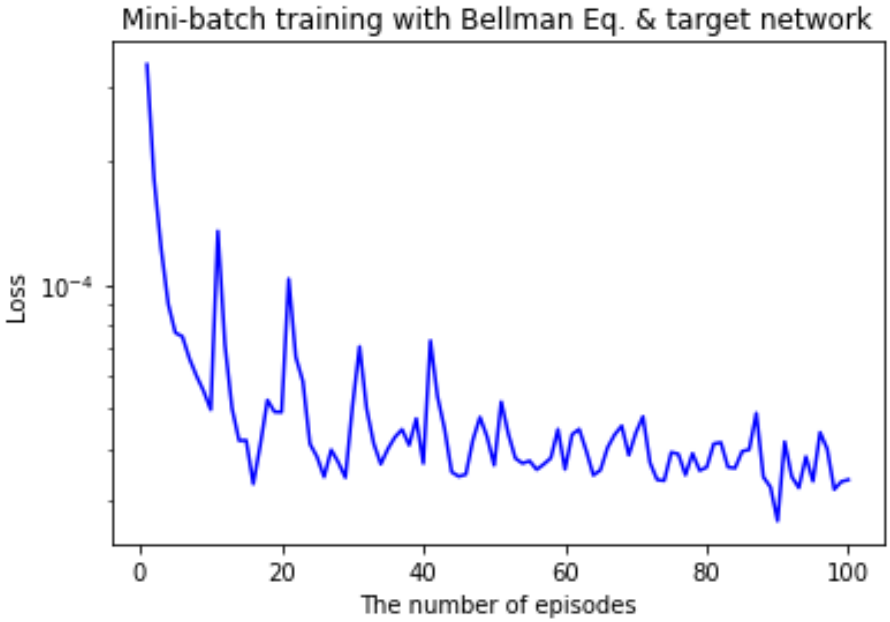

Figure 3 (b)

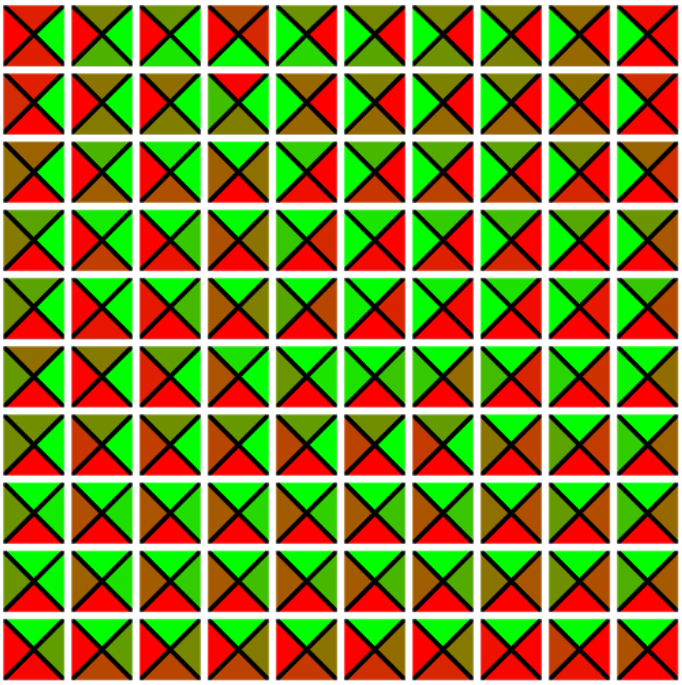

Figure 4 (a)

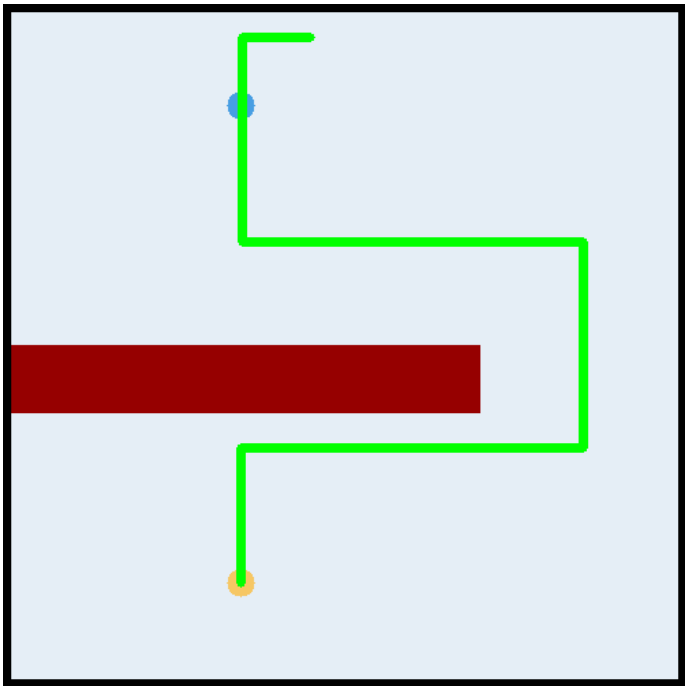

Figure 4 (b)

Description of Implementation for Part 2

The neural network structure is designed to have three hidden layers, each with 50 units, and ReLU function is applied to get the output of hidden layers (line 30 - 41). The optimizer uses *Adam* algorithm, with a learning rate $lr = 0.001$ (line 53). The double Q-learning method is implemented to achieve a stable learning process (line 80 - 109). In Bellman's Equation, the discount factor γ is 0.95 (line 103) for a visionary learning. The loss δ is calculated as the mean square value (line 106).

For the replay buffer, the maximum buffer length is set as 10,000 (line 213). Because the buffer is a double-end queue, if the number of transitions exceeds the buffer capacity, the oldest transitions will be automatically removed. The weight of a transition is calculated by (line 158 - 178):

$$w_i = \max|\delta| + \epsilon = \max_a |R + \gamma\, Q(S', a) - Q(S, A)| + 0.00001$$

When sampling in the replay buffer, the possibility of being sampled is determined by the transition's weight $p_i = \frac{w_i}{\sum_k w_k}$ (line 228 - 231), i.e. the Prioritised Experience replay method is applied here. The sample batch size increases every 20 steps, but with a maximum size of 128 (line 233 - 237). The weights of transitions will be updated with newly trained Q-network (line 181 - 203 & 255 - 259).

The initial episode length and agent step length are 640 and 0.02, respectively (line 264 - 302). An action is ε-greedily taken by the agent, but larger possibilities are assigned to moving up and down (line 134 - 138). To ensure a broad exploring at the beginning of learning process, the value of ε follows the equation below (line 121 - 129):

$$\varepsilon = \frac{1}{\textit{the number of episodes taken}} + \epsilon_{const} \; \textit{and} \; \varepsilon = \begin{cases} 1, \; if \; \varepsilon > 1 \\ 0, \; if \; \epsilon_{const} = 0 \end{cases}$$

ϵ_{const}, whose initial value is 0.54 (line 299), controls the minimum value of ε. The episode length and ϵ_{const} will be decreased by 10 and 0.01 every time the agent reaches the goal, until their values equal to 100 and 0 (line 331 - 341). (Values are only decreased once in the same episode) When episode length = 100 and $\epsilon_{const} = 0$, the Q-network will be tested every five episodes (line 344 - 348). If the agent reaches the goal within 100 steps with the testing network, the entire training will be stopped until programme finishes (line 363 - 366). This is to avoid the instability caused by over-training.

The reward $= 0.5 \times \sqrt{1 - \textit{distance to goal}}$ (line 351). This allows the agent to be more sensitive to the changes of *distance to goal* when it is closer to goal. A transition tuple is then created and added into the replay buffer. Afterwards, the Q-network will be trained with a mini batch (line 351 - 387). Every 10 episodes the target Q-network will be updated as the same as the regular Q-network (line 383 - 384).