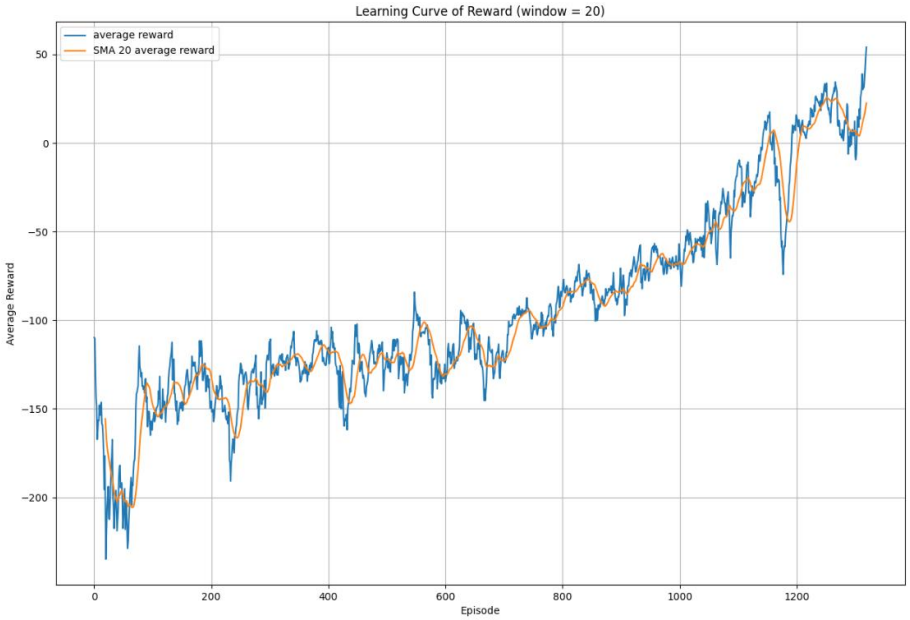
Q1：Model

網路架構都是使用助教提供的程式碼。

1. Policy Gradient：

由兩層全連接層組成，在選取$state_i$的$acton_i$時(make_action)，會利用機率分佈去挑機率最大的當 action，而$Reward_i$ 為$reward_i$ + gamma * $Reward_{i+1}$，這是因為不同時間的 reward 對當前時間 reward 的影響程度不同，而 model update 則是會對 loss 做梯度下降。

loss = $\sum_{i=0}^{n} - Reward_i * \log(\text{action\_prob}_i)$。



2. DQN ：

由三層卷基神經網路加兩層全連接層組成。

<u>Make action</u>：

使用 epsilon-greedy 去決定$step_i$的$acton_i$，一開始隨機生成$acton_i$，後面則由模型決定，而 epsilon 會隨 step 越大而漸漸變小。

Epsilon = Epsilon / (steps + 1)

<u>Update Model</u>：

Buffer 儲存很多的 transition($state_i$、$action_i$、$reward_i$、$state_{i+1}$)，每一次都隨機選出大小為 mini-batch 的 transition 處理。根據下列 loss 去做梯度下降：

$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a', w^-) - Q(s, a, w)\right)^2\right]$$

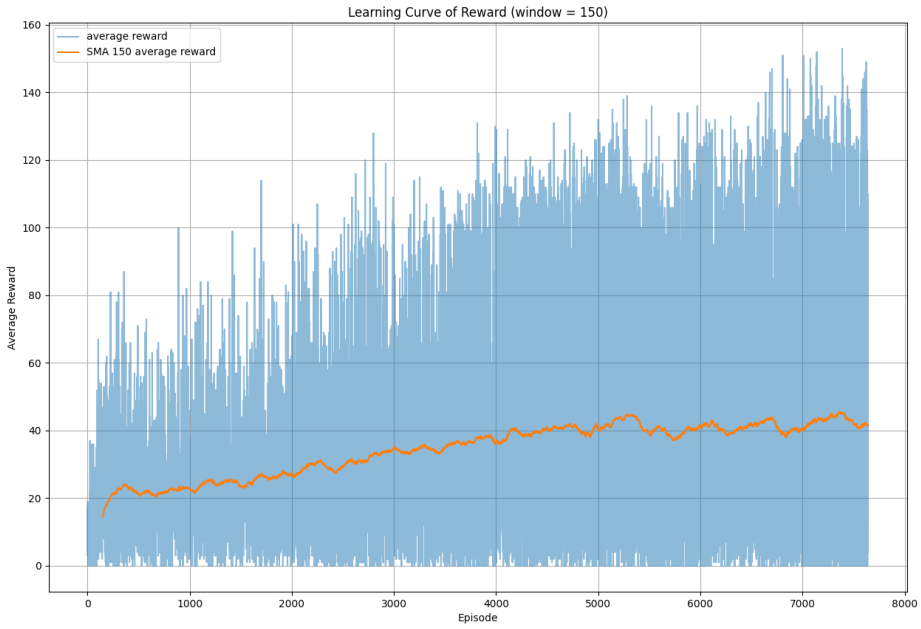
$\hat{Q}$為 target model，Q 為 online model，訓練時會固定$\hat{Q}$，訓練一段時間後會將$\hat{Q}$更新成 Q。

Step 1：使用 Q 去計算$state_i$做$action_i$的 online_reward

Step 2：使用$\hat{Q}$去計算$state_{i+1}$做所有 action，並選最大的當 target_reward

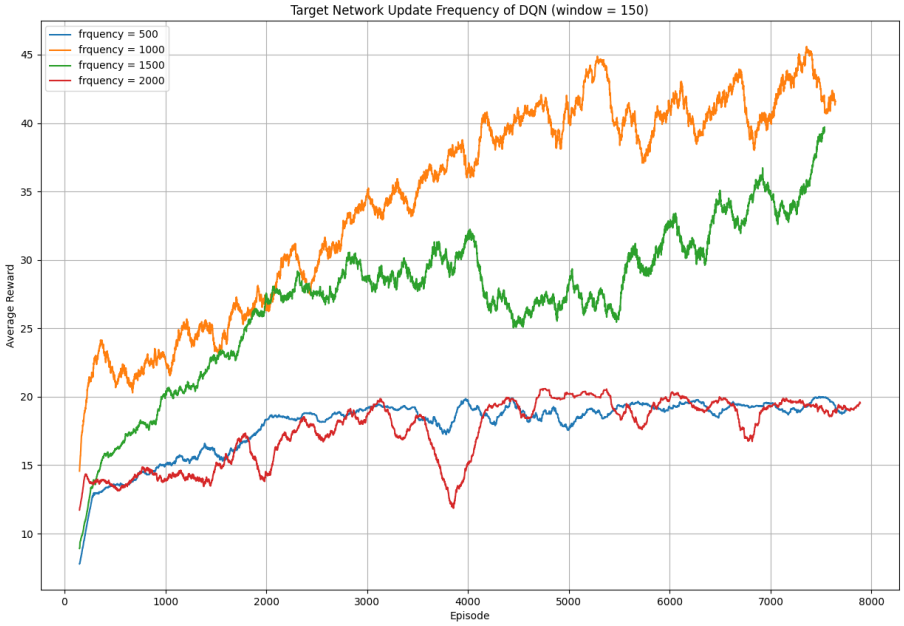
Step 3：loss = $((\text{gamma} * \text{target\_reward} + reward_i) - \text{online\_reward})^2$/ batch_size 並做梯度下降

Step 4：一段時間後(程式裡 target network update frequency = 1000)會使用 load_state_dict 將 online model 的 weight 更新到 target model。



Episode 大概在 7500 時就趨近平緩。

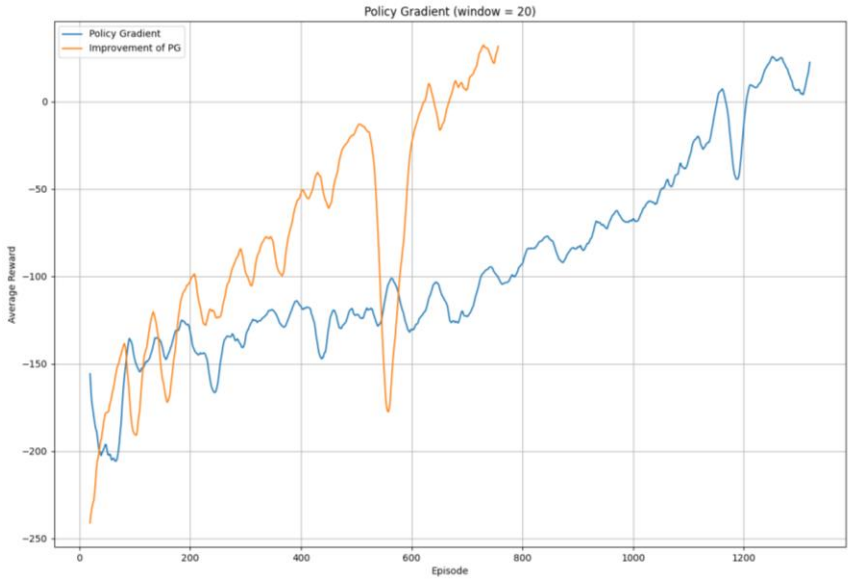
Q2：Hyperparameters of DQN



參數為 target network update frequency，探討 target network 更新頻率對於模型的學習有什麼影響，由上圖可以發現更新頻率在適當的狀況下模型才能跳脫區域最佳解，進而獲得更好的結果。更新頻率可以影響模型在震盪劇烈以及非常傾向某個方向時的狀態，如 reward 在非常好以及非常不好之間震盪時，延遲的更新頻率能將他們平均起來，使得趨勢便得平緩；但 reward 總是給予單一方向的回饋，而又有延遲的更新頻率時，會使得每一次 target 的更新都非常的劇烈。以實驗結果來看，較好的頻率為 1000(程式碼原始更新頻率)以及 1500。

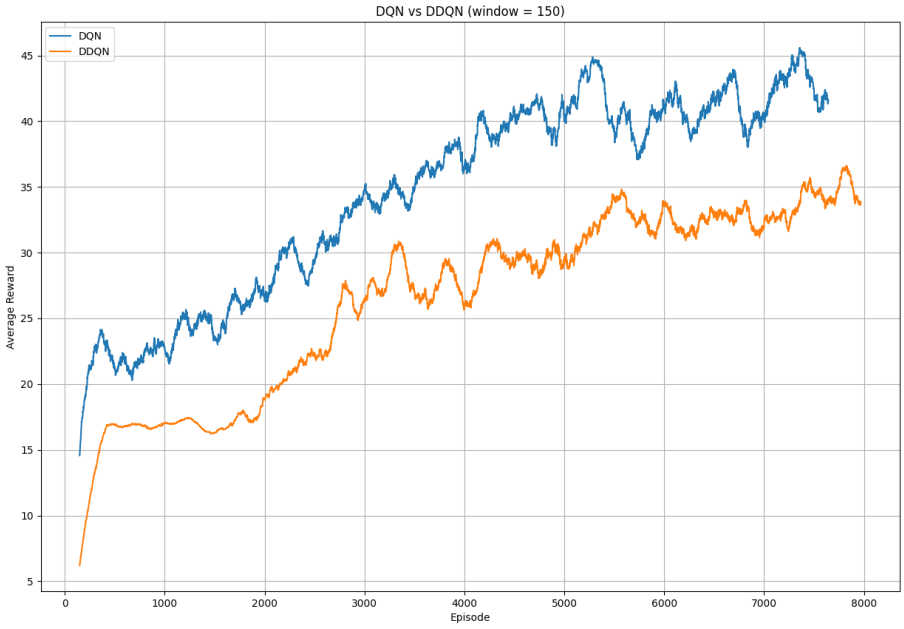
Q3：Improvements

1. PG with baseline



上圖藍線為原始 PG、橘線為 Improved PG。透過在計算 loss 的時候使用一個全局考量 baseline 參數(b=sum(R)/len(R))，使得 loss = sum(-(R_i-b) * log(action_prob))，能夠使 Variance 降低增加網路的收斂速度。如上圖所示，Inproved PG 在原始 PG 約一半的 Episode 數量便收斂完成。

2. Doubld DQN



上圖藍線為 DQN、橘線為 DDQN。此改進為將 target 的 Q 值計算由 target network Q 值最大值改為 online network Q 值最大的 action 代入 target network 得到，使得不會因為直接取最大值導致網路 Over Estimation。由於 DDQN 的特性使得網路較不會出現有 Over Estimation，因此上圖 DDQN

training curve 看起來會是比 DQN 稍低一些。