

1. Basic Performance

(1) Describe your Policy Gradient & DQN model

a. Policy Gradient :

Policy Gradient為基於神經網路的模型，輸入為observation，輸出便是action(也可為該action的發生機率)。Policy Gradient透過分析一盤遊戲中結果的好壞(贏/輸)來識別在一盤遊戲裡action的好壞，並將這盤贏的action視為是好的，並提升所做action的發生機率；反之亦然。這讓Policy Gradient得到一個target function讓其根據每盤當下的observation最大化它認為最好的一步。

b. Deep Q Learning, DQN :

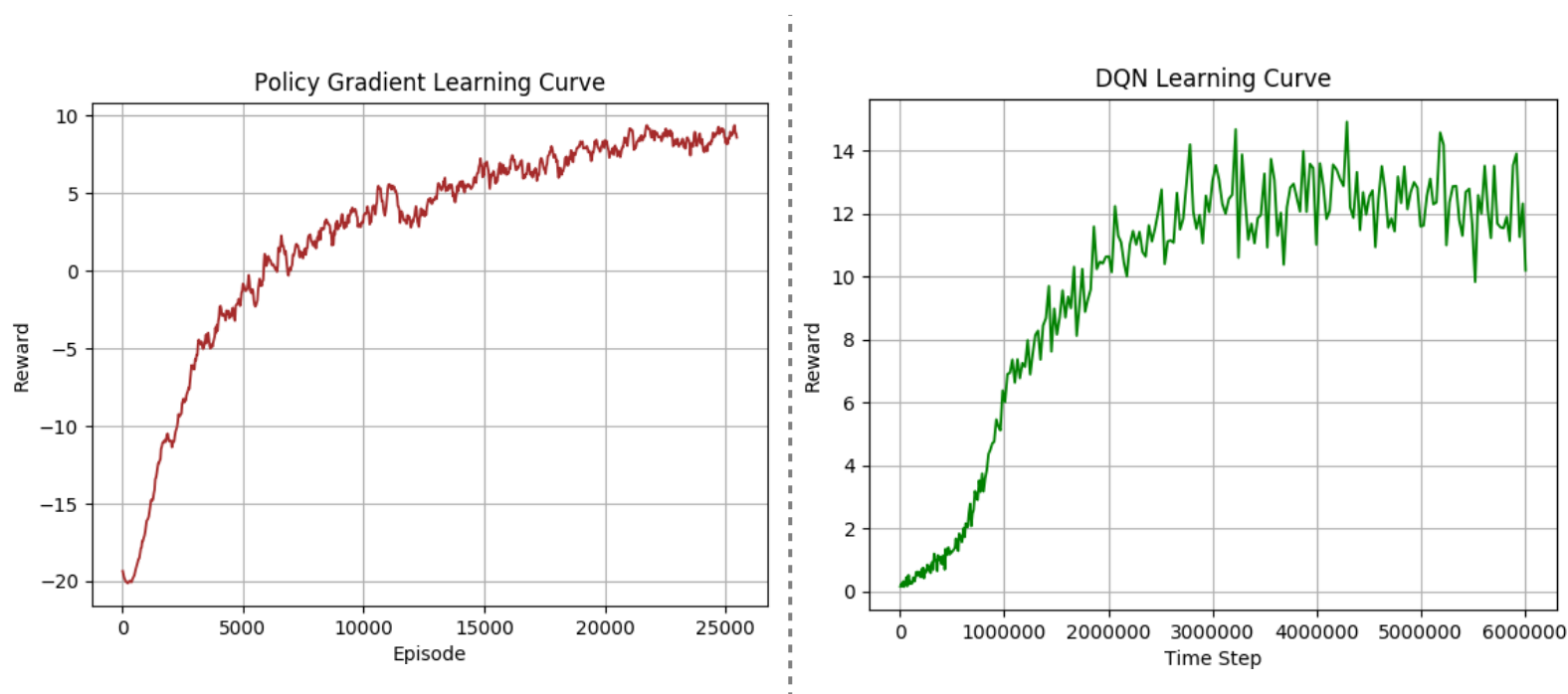
DQN具有以下3大結構特性：

- (1) Neural network：以一個target function來取代table。
- (2) Replay memory：存儲過往與環境互動的記憶，供之後訓練時隨機抽樣使用
- (3) Independent target network：解藕online network訓練時與其自身的相依性。

DQN與Policy Gradient不同的是DQN為state-based的算法，它並非直接決定要使用哪個action，而是透過分析一個observation來決定所有可能action的Q value(由target network評估)，再根據這些Q value來決定要使用哪一個action。

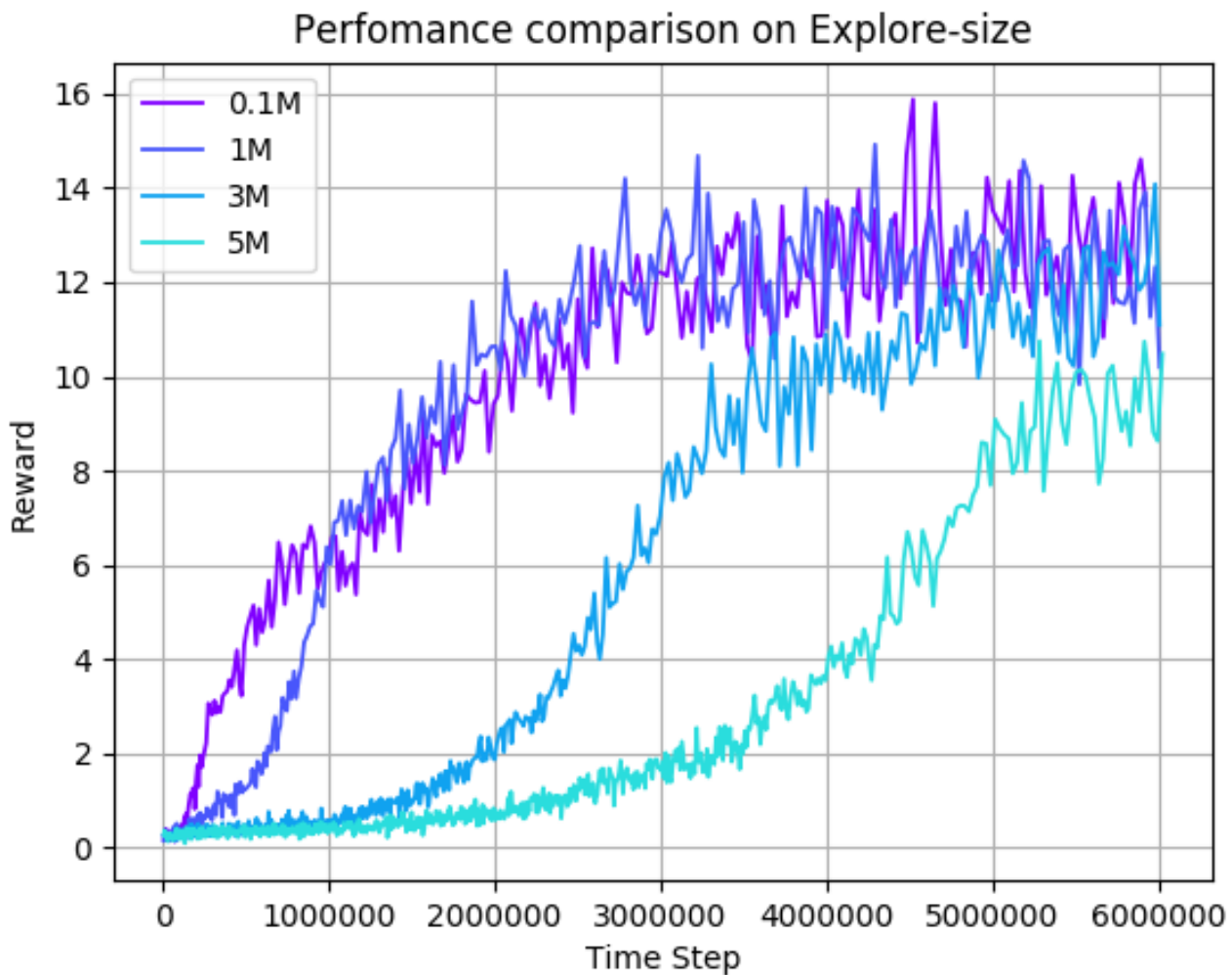
DQN透過降低現實(environment給予的feedback加上受其影響的未來observation的reward)與預測(當時所做得Q value)的差距來訓練模型。

(2), (3) Plot the learning curve of your Policy Gradient on Pong & DQN on Breakout



2. Experimenting with DQN hyper-parameters

(1) Plot all four learning curves in the same graph



(2) Explain why you choose this hyper-parameter and how it effect the results

我的explore rate是根據explore size線性下降，而我認為一個適當且低的explore size可以更有效率地訓練模型。

在圖中可以得知explore size調的越低，reward上升的越快，且在explore size為10萬的前提下，最終結果的reward並不會差距太大。

但是當explore size調高到一定程度，不僅訓練速度慢，且在最終結果上也有一定的差距。

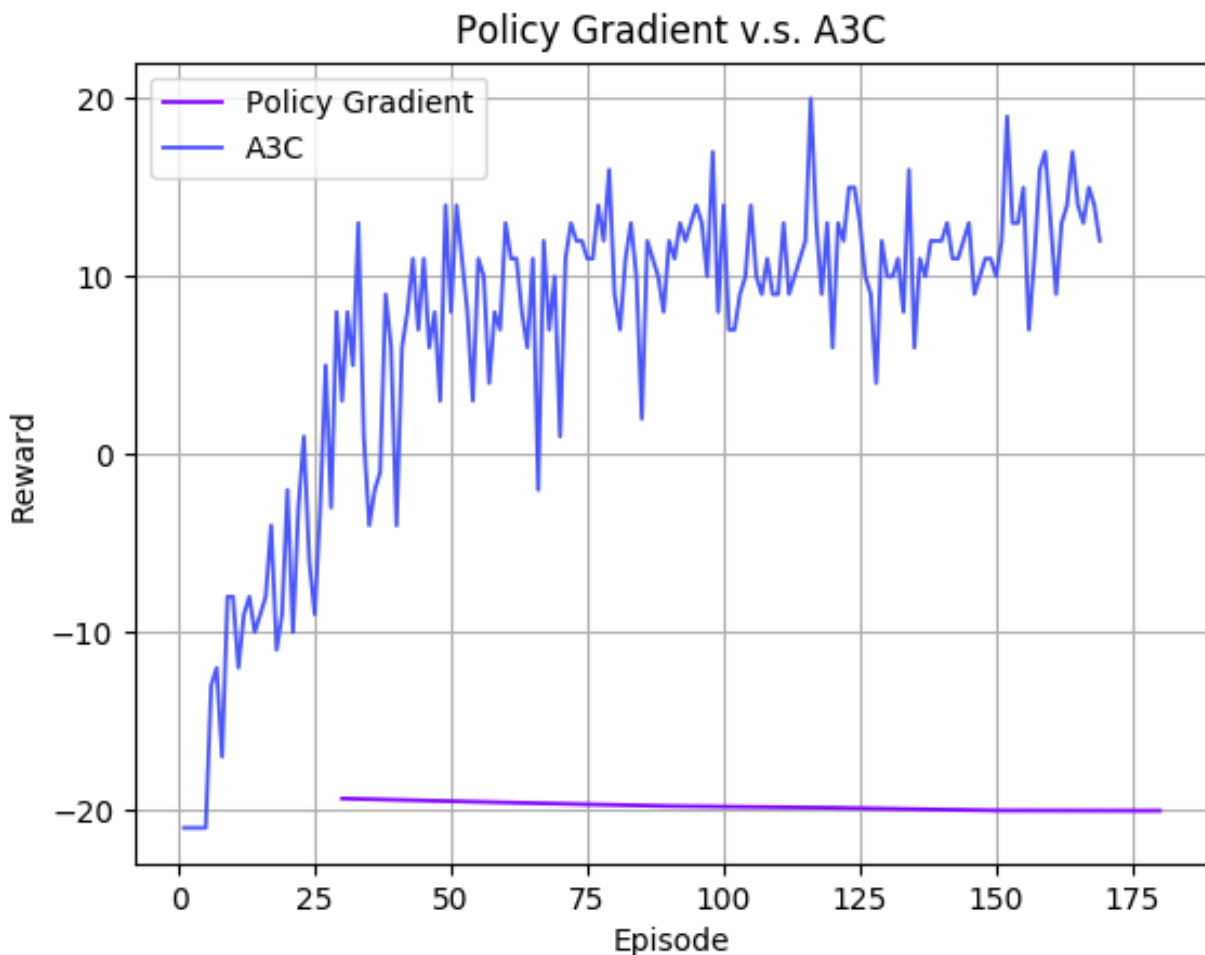
3. Bonus

(1) Improvements to Policy Gradient

a. Implement at least two improvements to Policy Gradient and describe why they can improve the performance

本作業實做Asynchronous Advantage Actor Critic, A3C，這會讓主機創建多個agent並行的環境，讓多個有同樣結構的agent同時運算，而這些並行的agent彼此並不互相干擾，並隨著訓練非同步地更新著主agent中的參數。簡單的說，這些從屬agent會從主agent那邊學習怎麼玩遊戲，再將各自玩遊戲遇到的feedback更新回主agent，而A3C採用的是輸出輸出動作的機率，即policy based，它在Q value的計算上使用了優勢(Advantage)來評估，優勢指的是在同個state下每個action相對於其他的action的值相差，舉例來說：假設狀態s的value為V，則優勢A即為 $Q(Q為在s的狀態下做出action的值) - V$ 。

b. Plot a graph to compare and analyze the results with and without the improvements



從上圖可以發現當A3C Train到約100個episode時，平均reward便已超過10，在同時期的policy gradient仍停留在-20得成績，由此可見A3C的Training效率非同小可。

(2) Improvements to DQN

a. Implement at least two improvements to DQN and describe why they can improve the performance

我實作了以下兩點：

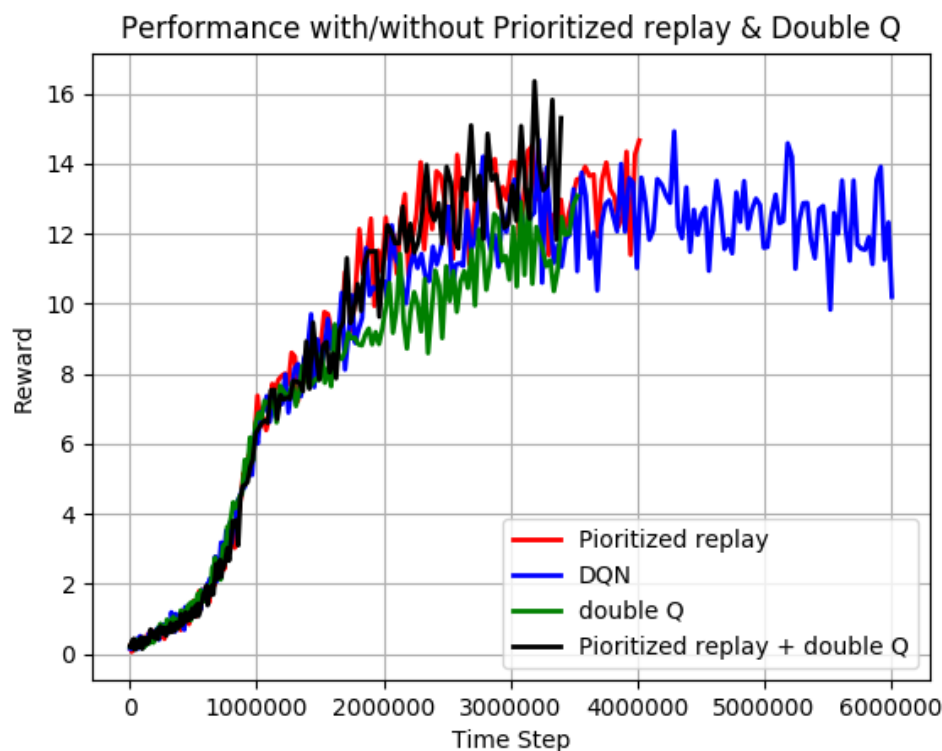
(1) Double Q-Learning

因為原本的神經網路(online network)在預測Qmax本來就有誤差，這可能會導致對於每個state(observation)的Q value的過度估計。因此 Double Q-Learning引入另一個獨立神經網路來盡可能消除這些最大誤差的影響。而獨立神經網路即target network。

(2) Prioritized Replay memory

原本抽樣訓練的方法為隨機抽樣，但我們最希望訓練到的應該是那些預測與現實落差最大的樣本(如：玩breakout順利打了好幾十球)，但這些樣本再所有樣本數裡的佔比較低，應此為了使我們能經常取得這些「珍稀」的樣本，我們將所有樣本按照預測與現實落差來進行排序，並調高這些「值得訓練的樣本」的抽樣機率。

b. Plot a graph to compare and analyze the results with and without the improvements



可以發現這些折線走向大多相同，但若仔細看，其實是有明顯的「小落差」。會造成落差如此之小我認為主要有兩個原因：

(1) Memory size只有1萬，因此原本sample到不錯的樣本的機率就不低(若不錯的樣本佔1/100，考慮到32的batch size，則每次訓練抽到不錯的樣本的機率超過1/3)，因此做prioritized replay memory效益感覺不高，若memory size變大很多，差距會更為明顯。

(2) Double Q-Learning，主要是用來避免因為過度樂觀地估計Q值，使得找到的並非最佳解。而我改進後的Double Q-Learning最後有達到跟最初的Q Learning相同的表現；或許這代表原本的模型便已經找到不錯的解了。且Double Q-Learning是在選擇多個action時的表現上會改善較多，但本次作業僅4個action，因此進步的幅度不會太大。