ADL hw5 Report
R04922139

Explain why DQN algorithm need these function and how you implement them:

1. **Experience replay (1%)**

爲了防止每一次的 input data 是有時間順序性而不是 data 彼此間獨立而造成每次更新都可能有 沒有辦法 converge 或 一直在來回震蕩 (因爲更新都是根據有關聯的data 來更新會造成 model 只會learn 到相關 data 而不是縱觀局面 而 overfit) 的情況，因此需要使用 experience replay 的方法即有一個 replay memory 的機制，將過去的歷史都記錄起來，而在更新的過程中去做 random sample (這樣就可以避免現在的 model 過於 fit 和現在 behaviour 過於相關的 更新，使data 變成 iid 更有統計上sgd 的意義)。

Implementation:

設計一個 長度爲 L 的 array 當作 replay memory，並將過去的 記錄 append 到這個array 上，更新時就使用 random sample 來選擇以哪些來做更新

2. **Target network (1%)**

爲防止傳統上 online learning 時 Q value 的時常更新造成 隨着時間的變化 policy 可能已經 無法收斂或是震蕩的情況，而在更新的時候使用了 前 C 次的 舊的參數的 network 來當作更新的 target (loss function 中的 y)，而這樣的更新和 experience replay 想要做的目的是一樣的，降低 model 的不穩定性。

Implementation:

每 C 次的 update 都會將 Q network 的所有參數 clone 到 target network 上，target network 和 Q network 的結構是一樣的

3. **epsilon greedy (1%)**

在 reinforcement learning 中很重要的一點就是 exploration X exploitation。epsilon greedy 的意思就是我有一定 1 - epsilon 的幾率會去做 random 的事情去探索我的 space，而 epsilon 的幾率則是我根據 Q network 出來最 greedy 的 policy。因此稱爲 epsilon greedy。

Implementation:

設定 epsilon 後，每次丟一個 random 值，如果那個 random 值 小於 1-epsilon，則隨機選 action，不然就照着 predict 出來的 argument max action 執行。

4. **clip reward (1%)**

因爲 Q learning 的更新會取決與 reward 來決定，而每個遊戲的 reward 可能都會很不一樣。爲了在每個遊戲都能通用，在計算 error 與 gradient 更能夠輕鬆的調整learning rate 以及 agent 可能沒有辦法區分哪個遊戲的多大程度的 reward 才是重要的這件事情，因此才做 clip reward 這個動作。

Implementation:

Atari 所 return 出來的 reward 會將其限制在 -1,1 之間，可以使用 max 與 min function 來做。

If a game can perform two actions at the same time, how will you solve this problem using DQN algorithm ?(there's no absolute answer) (2%)

如果 action set 的總數在可以接受的範圍，最簡單的做法便是將 兩兩 action 的組合concate 起來變成一個 action 來做 predict，而predict 後再去根據一開始怎麼組合的拆解出來便成爲 可以同時做 兩個 action。

另一個解法也許就是 使用 多個network 如果有多個 action 的話(一個 network 對應 一個 action，當然這裏會有一些其他的問題比如說reward 能不能分開拿到，或是他們之間的 correlation 要怎麼設計，有哪些 network 要一起 share 之類的問題)，最後再一起 sum 起來當作 loss function 去嘗試。