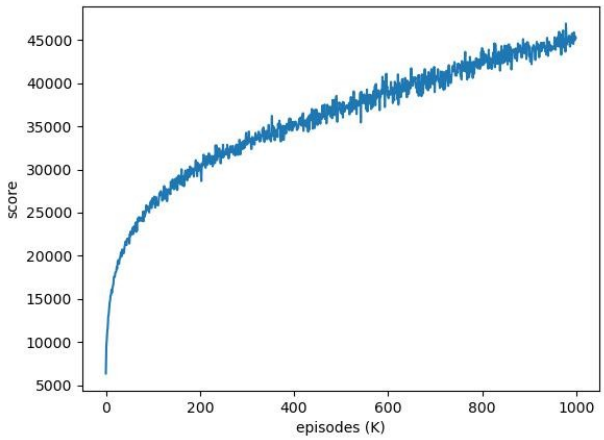
RL_LAB1_313552041_洪日昇
A plot shows scores (mean) of at least 100k training episodes (20%)



Bonus: (20%)

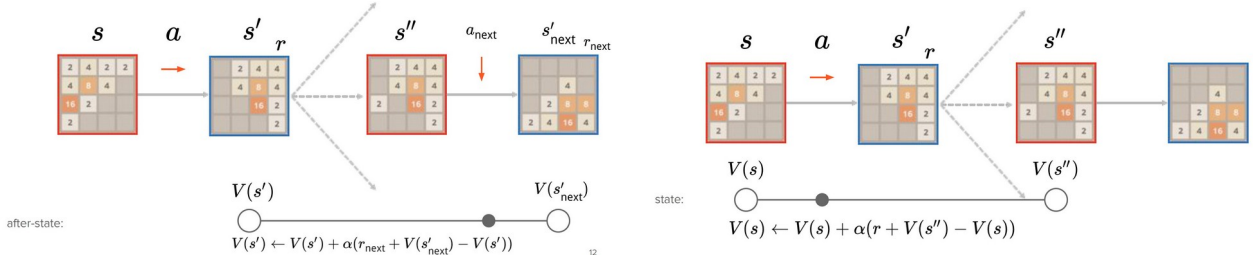
1.Describe the implementation and the usage of n-tuple network. (5%)

假設 2048 遊戲的盤面 16 格當中會出現 0-2048，則每一格有 12 種可能性，因此可能有 12^16 種甚至是更多的可能性，但是這樣子將會使記憶體無法負荷，使 AI 根本無法訓練，因此使用n-tuple network 紀錄盤面的特徵，不用把整盤都紀錄下來，在估計盤面時也只對選取好的特徵更新數值，此外會將選取好的特徵找到八種同構的圖面，這樣子若出現同構的盤面時可以確定估計值會是相同的，盤面也不會有被忽略的地方。

2.Explain the mechanism of TD(0). (5%)

TD(0)與 MC 都是利用經驗來學習，相較於 MC 需要做完整個 episode 才將回傳的數值做更新，TD(0)的特色是不需要等待整個做完才更新數值，可以在每一個狀態點都做更新，MC 的 episode 若非常長，則中間可能會很多複雜的 reward 混雜，也因此 MC 的 variance 較大，TD(0)則會存在 bias 問題，因為 TD(0)用的是 next 來更新而不是真正的結果。

3.Describe your implementation in detail including action selection and TD-backup diagram. (10%)



3.1TD-backup of V(after-state) and action selection

V(s') ← V(s') + α(rnext + V(s'next) − V(s'))
在 after-state 中 error 會是 next action 的 reward 加上 V(s'next) 扣掉 V(s')，best_action 只要選擇當下 reward+V(after-state)最大的值即可，不用考慮隨機 popup 2 或 4 的情形。

3.2TD-backup of V(state) and action selection

$V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$
在(before-)state 中 error 會是 action 的 reward 加上下一個 before-state 的數值扣掉當下 before-state 的數值，best_action 要注意 popup 的所有可能性，將所有 board 的數值乘上 2、4 出現的機率，就可以得到現在的數值

3.3 implementation

由於有特別要求”Update V(state), not V(after-state).That is, you need to use the information of P(popup tile 2) = 0.9 and P(popuptile 4) = 0.1 in your code.”，我的 select_best_move 中檢查 board 中的所有 16 個格子，若為空則放入 2 與 4 以及分別將兩項的 estimate 乘上 0.9、0.1 的出現機率，再加總，最後比較出最大的並回傳。

在 update_episode 中，一開始的 target 為 0，在 terminal state 沒辦法再走下去了，透過下列計算公式 $V(s) \leftarrow V(s) + \alpha(r + V(s'') - Va(s))$，我們可以得知 error 為 $r + V(s'') - Va(s)$，更新 before state 的值，並且存在 target 中，提供下一次使用，最終將所有數值更新完畢。