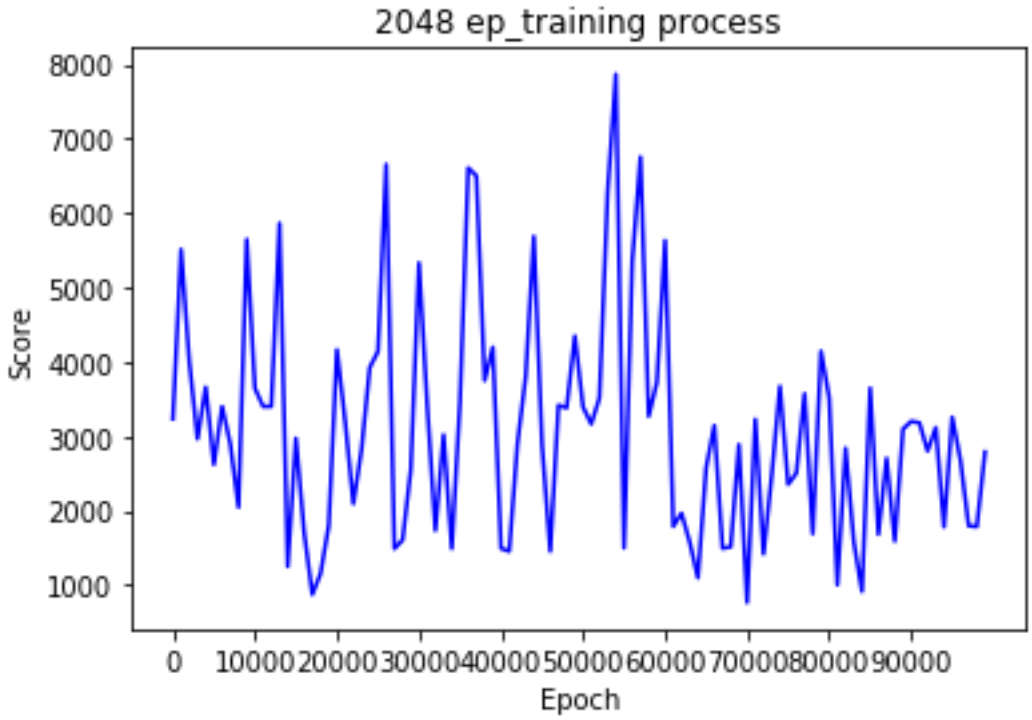
Lab7 - 2048 TD Learning

1. A plot shows episode scores of at least 100,000 training episodes



2. Explain the mechanism of TD(0)

TD learning 結合了 dynamic programming 和 Monte Carlo approaches 的優點，首先是 dynamic programming 的部分，它能在優化在 initial state 的 value function 時，利用下一個狀態的期望值來更新上一步的模型，不需要等到最終的 outcome 出來，這個過程稱為 bootstraping。另外在 Monte Carlo approaches 的部分，我們不需要 environment dynamics 的模型而直接可以從經驗中學習。

TD 和 Monte Carlo approaches 都使用經驗來解決預測問題，給定服從規則 π 的一些 policy，2 種方法皆可以更新 policy 中的每一個非終止狀態 S_t 的 v_π，但是 Monte Carlo approaches 要等到 return 知道之後才將其設為是 $V(S_t)$的目標值，表示為:

$$V(S_t) \Leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

其中 G_t 代表的是時間 t 之後的真實 return，alpha 是固定的 step-size 參數，可以將這種方法稱為是 constant-alpha_MC，Monte Carlo approaches 必須等到 episode 結束之後才能決定 $V(S_t)$的增加。

與 Monte Carlo approaches 不同的是 TD learning 只需等到下一個 time step 即可，就是在時刻 t+1 可以立即形成一個 target，並使用觀測到的 reward R_{t+1} 和估計的 $V(S_{t+1})$進行更新，最簡單的方法稱為 TD (0)，其更新方法為:

$$V(S_t) \Leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

若比較 Monte Carlo approaches、dynamic programming、TD learning 的目標值，

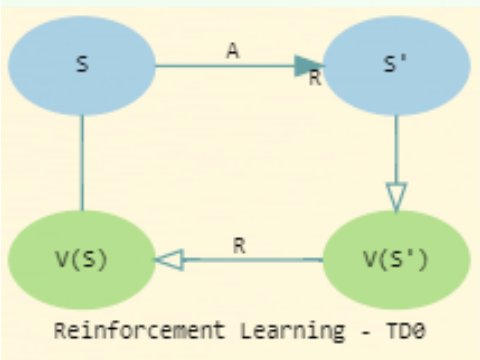
Monte Carlo approaches: $v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$

Dynamic programming: $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$

TD learning 因為 $v_\pi(S_{t+1})$ 未知，因此利用當前的估計值 $V(S_{t+1})$ 來代替，

其中 Monte Carlo approaches 和 TD (或 DP)更新的更新可由下圖來看:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2}|S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]
\end{aligned}
$$



Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

在 TD 學習中還有一個重要的概念叫 TD error，以 δ_t 表示，表示的就是在該時刻的估計誤差，就是在 $V(S_t)$的誤差:

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

另外，Monte Carlo error 可以寫作是一系列 TE errors 的和：

$$
\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma(G_{t+1} - V(S_{t+1})) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - V(S_{t+2})) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t+1}\delta_{T-t+1} + \gamma^{T-t}(G_T - V(S_T)) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t+1}\delta_{T-t+1} + \gamma^{T-t}(0 - 0) \\
&= \sum_{k=0}^{T-t+1} \gamma^k \delta_{t+k}
\end{aligned}
$$

3. Describe how to train and use a V(state) network

若是要訓練 V(state) network，此時 Q-table 所記的 state 為做完 action 且新 tile 隨機跑出來之後的狀態，因此 Q-value 輸出的 output 的參數是新 tile 跑出後的狀態以及該次動作 action。至於在 pseudocode 方面:

TD(0)-state

function EVALUATE(s, a)
  $s', r \leftarrow$ COMPUTE AFTERSTATE(s, a)
  $S'' \leftarrow$ ALL POSSIBLE NEXT STATES(s')
  return $r + \Sigma_{s'' \in S''} P(s, a, s'') V(s'')$

function LEARN EVALUATION(s, a, r, s', s'')
  $V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$

V(state) network 一開始要先計算出 after-state 及其 reward，還要再藉由 after-state 去推算每一種有下一個可能的 state 的發生機率是多少。第一個 evaluation function，除了要 return 下一個 state 的 value function，因為中間隔一層不確定性的 after-state，所以在下一個 state 的 value function 前還要在乘上該 state 的發生機率，最後更新 Q 值就是很正常的依照已經推倒過後的 TD(0)公式去做更新。

4. Describe how to train and use a V(after-state) network

在這次的 lab 裡面，我是使用 V(after-state) network 來實作，after-state 就是指 board 在執行 action 後但在新 tile 尚未跑出前的狀態，因此 Q-value 輸出的 output 的參數是新 tile 跑出前的狀態以及該次動作 action。此外，在更新 TD-Afterstate Network

TD(0)-afterstate

function EVALUATE(s, a)
  $s', r \leftarrow$ COMPUTE AFTERSTATE(s, a)
  return $r + V(s')$

function LEARN EVALUATION(s, a, r, s', s'')
  $a_{next} \leftarrow \underset{a' \in A(s'')}{\operatorname{argmax}} EVALUATE(s'', a')$
  $s'_{next}, r_{next} \leftarrow COMPUTE\ \ AFTERSTATE(s'', a_{next})$
  $V(s') \leftarrow V(s') + \alpha(r_{next} + V(s'_{next}) - V(s'))$

V(after-state) network 比 V(state) network 要簡單上許多，因為少了計算機率矩陣的關係，V(after-state) network 的更新主要是先計算出 after-state 及其 reward。第一個 evaluation function，只要 return 下一個 after-state 的 value function 即可，最後更新 Q 值就是去用當前 after-state 和 action 下 Q-table 的最大值，去和前一次 after-state 的 value function 以及 reward 去做在之前已經推倒過後的 TD(0)公式去做更新。

5. Describe how the code work (the whole code)

在這整個程式碼中，主要分成兩部分，第一個部分是建構環境，第二個是做運算，建構環

境的部分不用 gym 函式庫的原因是之前助教說可能會有運算太久的問題，所以提供給我們他們實驗室自己手刻的 2048 環境，在這個 lab 主要是以 Q-Learning 去做學習，但是我發現我並沒有做得很好，並且因為 Q-table 是單純地將每個狀態都記住，不像 DQN 有 generalization 的特性，所以計算到一半也發生 memory 爆炸的問題，所以我最後就改用 agent 只往 reward 較大的這個方式去執行 action，步驟就是先有一個處於 initial state 的 board，之後在每個 board 的 after-state 狀態之下，計算出 agent 往哪個方向移動，會有較大的 reward，就往該方向執行 action，最後 terminate 的時候是當無論哪個方向的 reward 皆為-1，也就是死盤的時候，就可以重新 reset 一個全新的 board 了。

6. More you want to say

在這次的 lab 中，TD learning 的理論相對於之前的深度學習網絡其實並不難理解，但是困難的點是在於 reinforcement 在網路上的資源幾乎都是搭配 Neuron Network 去做運用，此外，網路上在玩小遊戲的 reinforcement learning algorithm 主要都是搭配 gym 函式庫，而這次 2048 主要卻是以手刻為主，以上是我覺得這次的 lab 稍難的原因。

7. Strength (Python version)

Learning score table

Maxium scores: 28495

Score	Frequency	Ratio
0 ~ 1000	2435	0.0244
1000 ~ 2000	25245	0.2524
2000 ~ 3000	23957	0.2396
3000 ~ 4000	28495	0.285
4000 ~	19868	0.1987