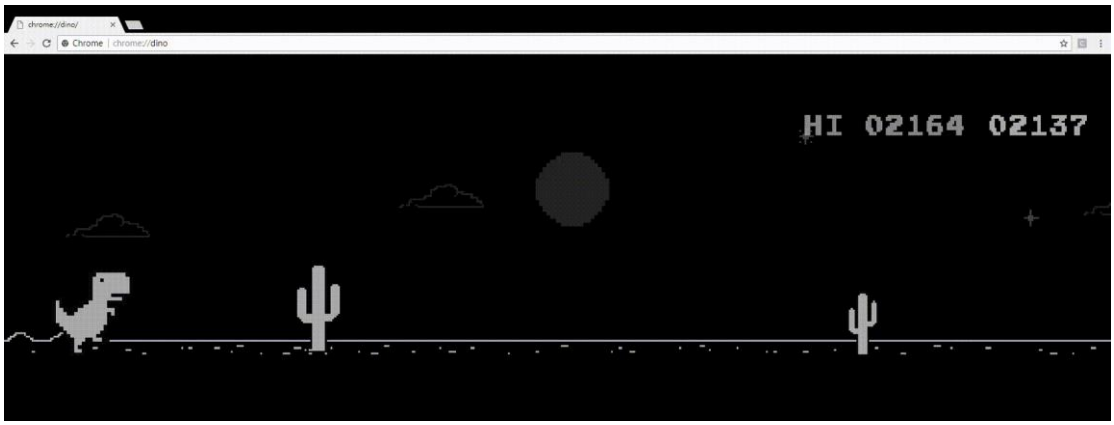
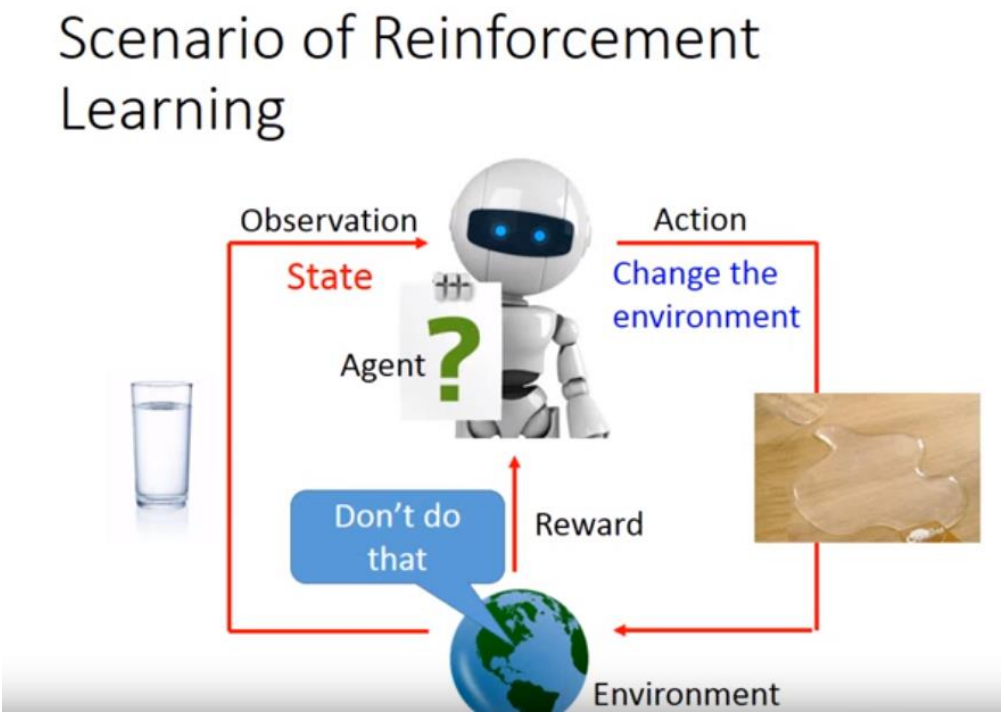
Reinforcement Learning for Dino_run



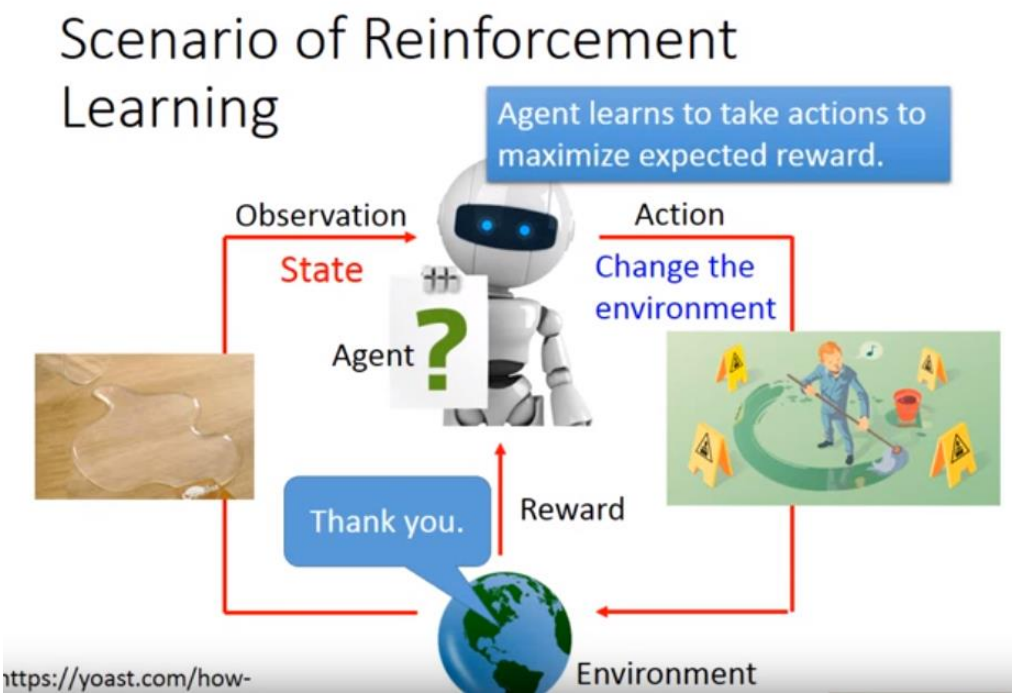
強化學習概述

環境描述

1.觀察 2.選擇行為 3.行動 4.獲得 reward



以示意圖而言 **Agent** 觀察環境，並且行動打翻水杯，獲得一個負向的 **reward**。而這次的動作會影響環境，下一個 **State** 會從打翻水杯後開始，若此時選取動作:清理打翻> > 的水杯，則會獲得一個正向的 **reward**。



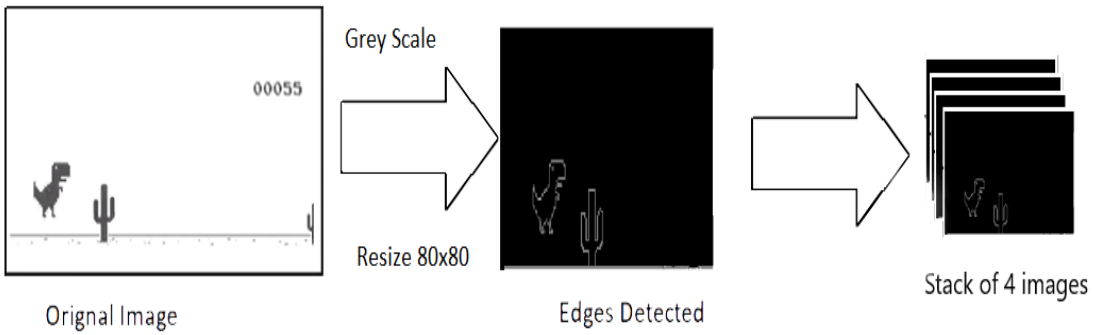
訓練方式

首先讓 Agent 在空間中探索，並且記錄每一次的探索經驗(S,A,R,St+1,Terminal)，狀態(time=t)、動作、報酬、動作後狀態(time=t+1)、是否終止。反覆重複 N 次作 > 為訓練資料，其中每次的動作是由 Q-Table 來決定的，依照環境只要生存就會提高獎勵，最後在利用這些經驗進行監督是學習，最佳化 Q-Table 使得 Agent 可以傾向於得到最多的 reward。

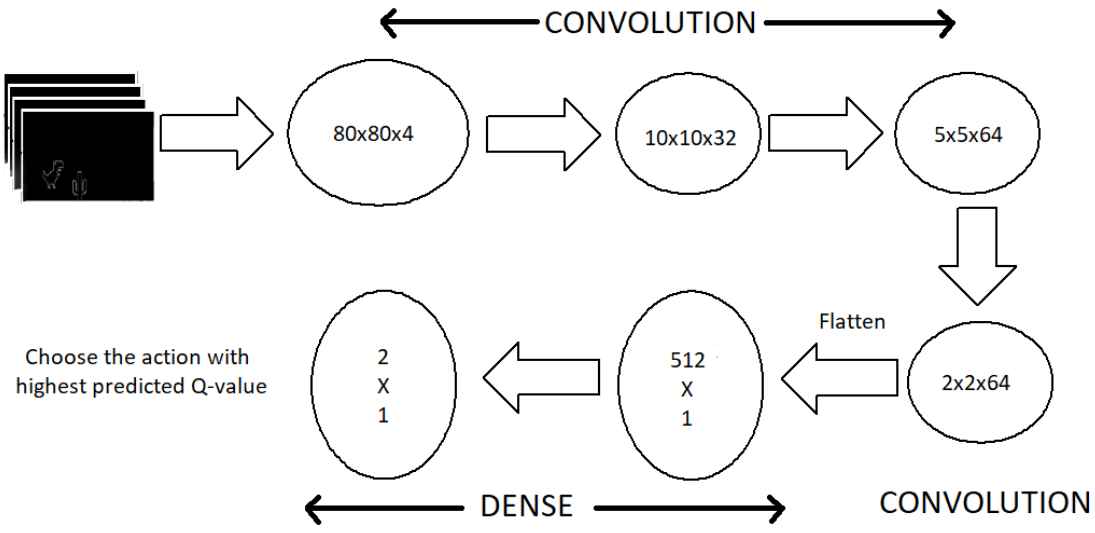
$$Q = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

應用 Dino-run

Q-Table 的設置，由於狀態是由影像構成的，無法一一窮舉成 Q-Table，故使用類神經網路來幫助我們達成 Q-Table 的近似，又影像為連續的相片，我們每次擷取連續的四張為一組 State Input。細節上我們將原圖進行 ROI(region of interest)擷取出小恐龍前方的障礙物的區塊，因為現實讓我們只在乎障礙物與畫面左邊界的距離，到達某個距離便是跳躍的關鍵。(80x80x4)



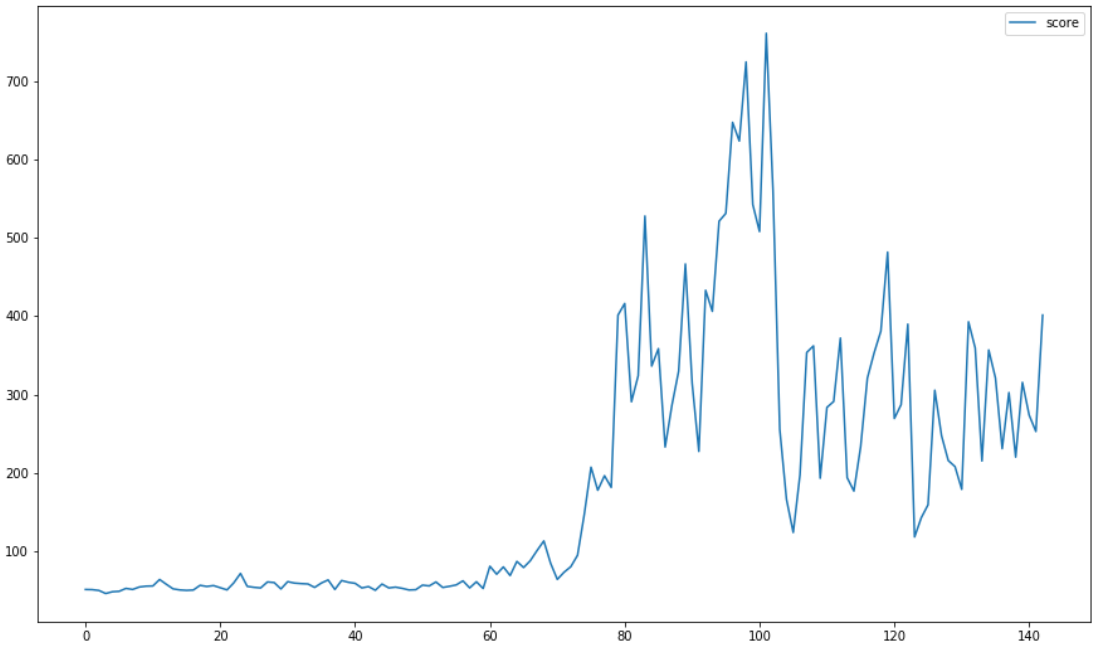
取得 80x80x4 的 State 後先進行 convolution 與 max polling，在經過 flatten 與全連接層，最後輸出兩個數字，用來表示 Q 值，我們通常會選擇大者為下一個採取的 Ation。



訓練細節

一開始讓小恐龍進行 100 次的觀察，獲得 100 組的(S,A,R,St+1,Terminal)。之後每觀察一組資料後便會從 Replay Buffer 隨機抽出 32 組資料出來訓練。具體獎勵生存的方式為 0.1+γ xMax(Q1,Q2)，γ =0.99，用來解決 Credit Assignment 的問題:我們不知道目前的 reward 是否確實是因為上一個 Action 完全造成的。因此設定 γ 。另外還有設定隨著時間遞減的隨機動作因子 epsilon，如此一來可以增加探索的可能性。

成果與分析



上圖為 moving average=15，可以看出雖然在中間部分成績有明顯的突出，但是右段卻下跌，我推測模型的不穩定性是因為 replay buffer 的資料新增與刪減的關係，記憶體的保存觀察組數為五萬組，多餘的會從最舊的開始移除，按照機率來說先進 queue 的資料會訓練的比較多次，因此在這個時候容易有過度擬合的現象，因此資料的 fitting 會較明顯，而隨著時間 push 新資料，模型需要擬合新資料，便會造成分數震盪。這種狀況我猜想只有在資料量不足的情況下才會發生，因為我是使用 CPU 訓練，礙於時間關係沒辦法完整的分析，若是以更大的尺度來看的話，整體的成績是成長的。最後成績最高分達到 3 千 7 百分。

心得

最後，我認為這次的 DQN 是沒有效率的，我自己親自玩過，輕鬆就可以達到 2 千分，礙於耐心的關係沒有繼續。這個網路我訓練 10 或 20 小時最高分也只有三千多。若我使用原始的計算機視覺依照障礙物的距離選擇動作，我想會做得更好。而我猜想結果不好的原因是 value base 策略在更新 reward 的時候，感覺就像是:你在狀況 S 下採取動作 A 的話，你"接不接近"獲得高 Q 值的 State?如果接近，那麼你的"所有"action 的 Q 值都會提高，這種現象是不合理的，這會使得訓練中出現許多不穩定的阻礙，因為有時候不一定是 Agent 採取的策略讓而得到 reward，而可能是"剛好"接近高 Q 值得 state。若是增加訓練次數，不斷有新的資料加入，我想最後結果會漸漸收斂得很好。

問題與討論

1. What kind of RL algorithms did you use? value-based, policy-based, model-based? why?
2. This algorithm is off-policy or on-policy? why?
3. How does your algorithm solve the correlation problem in the same MDP?

回答 1: value-based algorithm, DQN 使用神經網路作為 Q-table 的依據，在這邊我們輸出只有兩個(不跳與跳)，而神經網路輸出大值為即將採取的動作，故只比較 Q-value 大小，為 values-based。 回答 2:這是一個 off-policy 的 algorithms，因為在選擇動作時加入了一個隨機動作的可能性，已達到更好的探索空間。 回答 3:這裡處理資料相依的問題是使用固定 size 的 replay buffer 加上隨機抽樣的批次訓練，當 buffer size 大於 5 萬時便會 pop 最舊的資料。

更詳細解釋請參考程式碼: https://github.com/baker12355/Dino_run

參考自:https://blog.paperspace.com/dino-run/