ORGANIZATION

* 1. Introduction
  2. Implementation
     1. Bellman error for DQN
     2. Modify DQN to Double DQN
     3. Implement the memory buffer for PER
     4. Modify the 1-step return to multi-step return
     5. Weight & Bias
     6. Each Task implementation
  3. Analysis and discussions
     1. Training curves
        1. Task 1
        2. Task 2
        3. Task 3
     2. efficiency with enhancements

1. Introduction (to report)

本實驗實作DQN，先以狀態較為簡單的CartPole環境作為測試，並透過Bellman Error來更新模型的梯度。接著將相同的架構應用在Atari遊戲Pong上，並進一步加入Prioritized Experience Replay以及Multi-step Learning機制，以更有效率地挑選訓練樣本，並透過一次傳遞多個步驟的reward來加速模型的收斂和效能提升。

1. Implementation
   1. Bellman error for DQN
      1. Task 1、Task 2

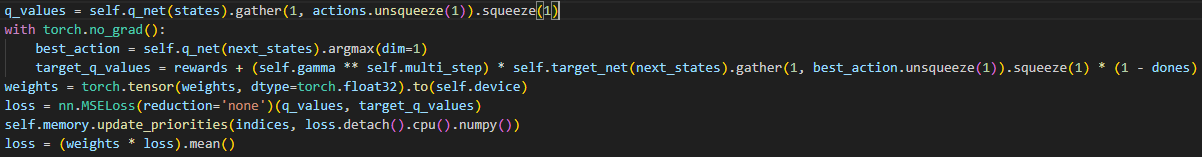






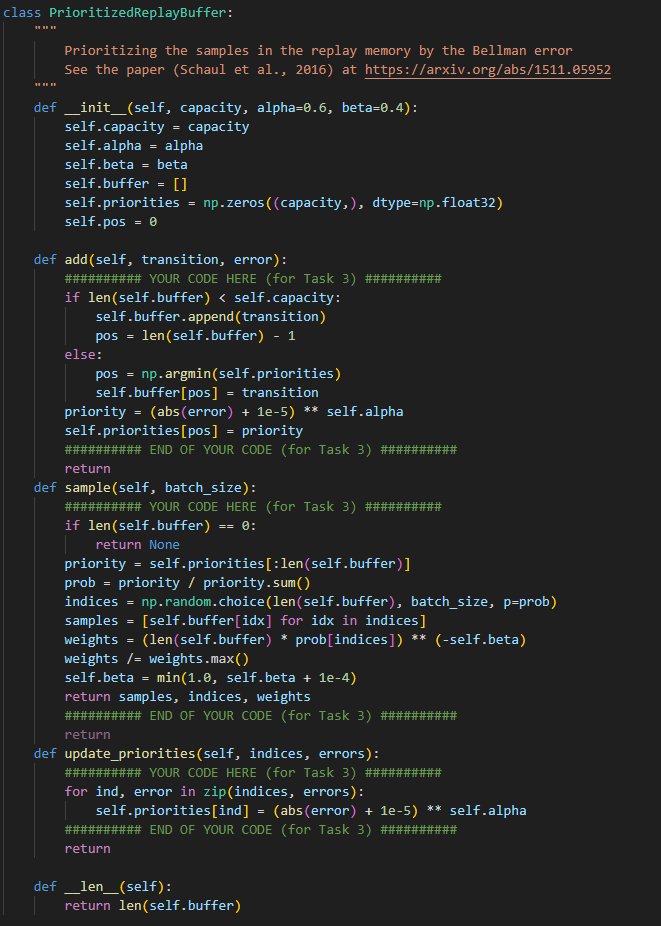
Bellman error的計算公式是希望q\_values輸出的值盡可能的靠近target\_q\_values，而target\_q\_values是由目前狀態與環境互動所得到的reward加上gamma乘以下一個狀態執行action所能得到的最大Q值所得來的。然後將q\_values和target\_q\_values計算MSE就可以得到這個batch size的Bellman error。

* 1. Modify DQN to Double DQN



與DQN不同的是，target\_q\_values中不再採用target\_net網路對於next\_states的最大Q值，而是將其更改為next\_states經由q\_net網路所決定Q值的最大動作，交給target\_net去得出的Q值。由此更改就能夠讓網路不會高估Q值造成成效不好。

* 1. Implement the memory buffer for PER



* + 1. Add

在新增當中，如果資料大小尚未達到capacity，就會直接加進去，而如果大小已滿，則會將最小error的那筆資料做替換(和一般的PER實作不同)。

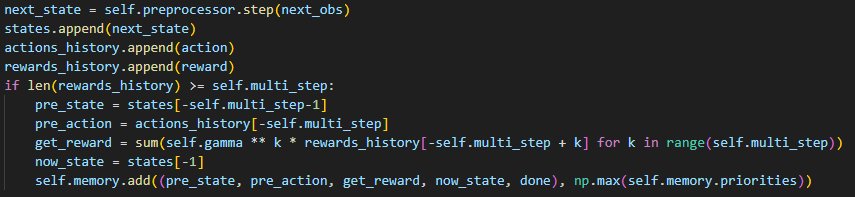
* + 1. Sample

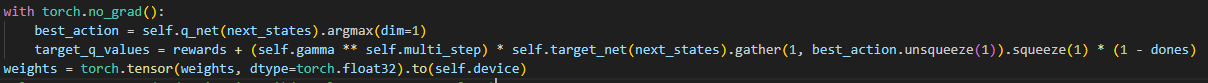
根據priority(已做alpha次方)的比例挑選資料，並計算其weights作為計算loss時候所需要的資訊。

* + 1. Update

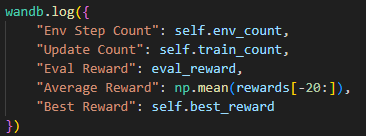
將indices的資料中更新error，為了避免error為0導致sample不到該筆資料，因此會加上一個eps確保該資料能夠有小機率被選取到。

* 1. Modify the 1-step return to multi-step return



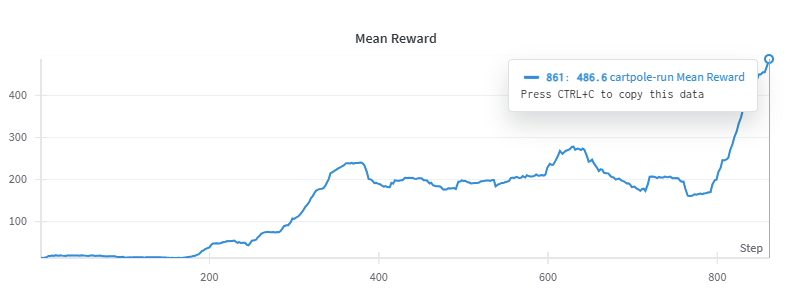
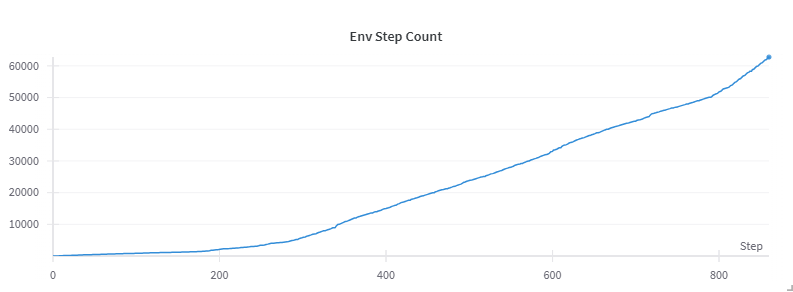


在n-step當中，不同於1-step，實作上會儲存過去的狀態、動作和reward，再將前n步的所有discount reward加總，作為(s,a,r,s)的資料，計算loss的時候，就會同時考慮第1~n的reward以及n+1的估計Q值。

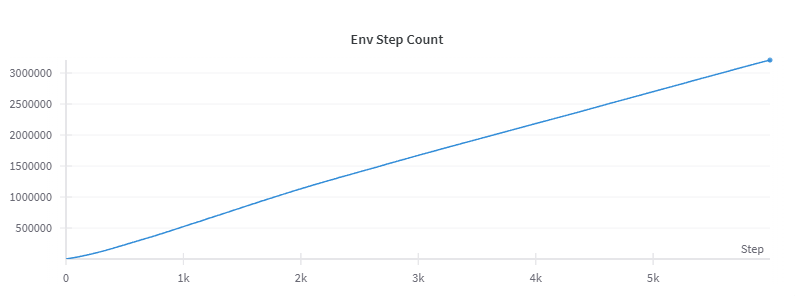
* 1. Weight & Bias 

在使用wandb當中，除了範例code所記錄的各項數值，本次實作還額外新增的Average Reward和Best Reward資料，這些是在evaluation階段紀錄的，根據evaluate的結果紀錄rewards，並且將最新的20場平均reward顯示出來，以符合本次lab的評估標準。

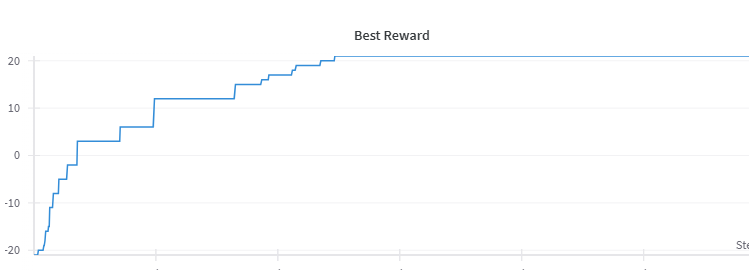
1. Analysis and discussions
   1. Training curves
      1. Task 1



* + 1. Task 2







* + 1. Task 3
  1. 12