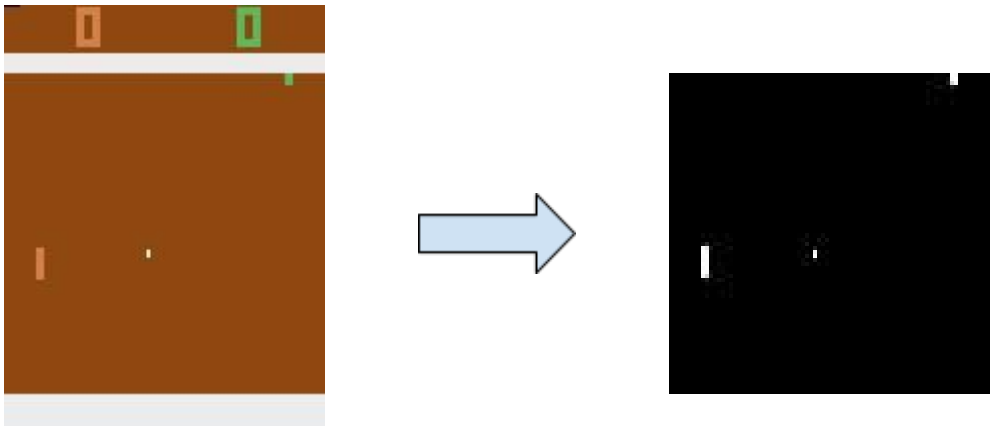
MLDS Homework 4 Report

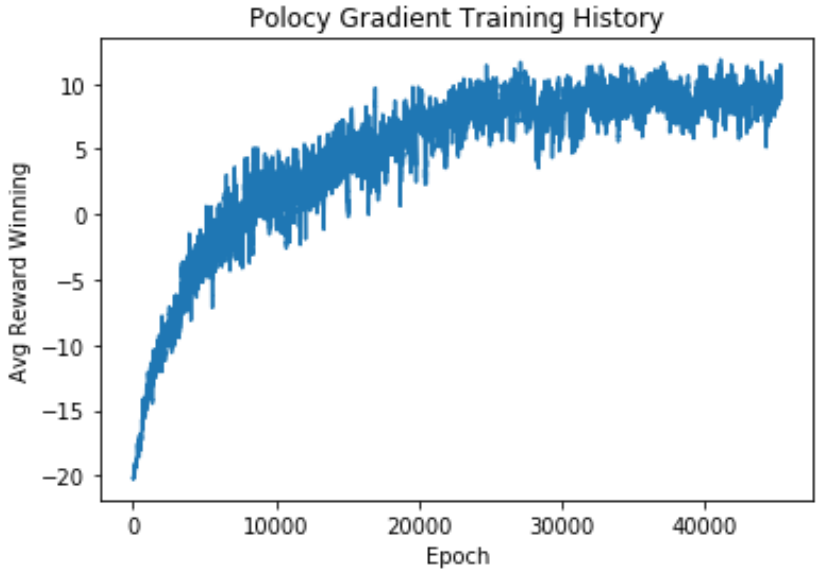
4-1

Describe your Policy Gradient model

我們使用的 Policy gradient model 有兩層的 Linear Model， 第一層吃進 Reshape 過的環境後輸出 256 維，通過 relu 後再經過第二層的 Liner 輸出為 3 維的 Action。將input進到network之前，我們會先對畫面做preprocessing，裁切成80*80的黑白圖片，如下圖所示：



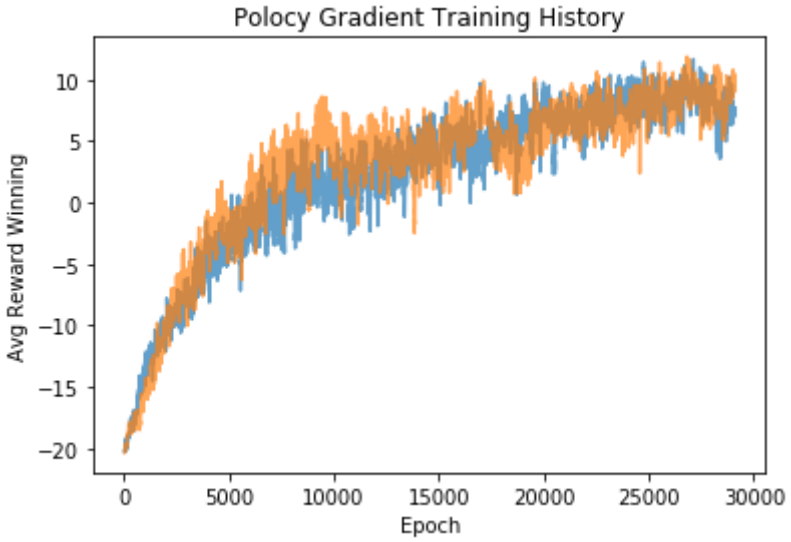
Plot the learning curve to show the performance of your Policy Gradient on Pong



最後三十個 Episode 平均分差為 +9.3。

Implement 1 improvement method on page 8

- **Describe your tips for improvement**
  將原本 Policy Gradient 的損失函數改成 PPO2 的算法，並且設置 Epsilon 為 0.2。
- **Learning curve**



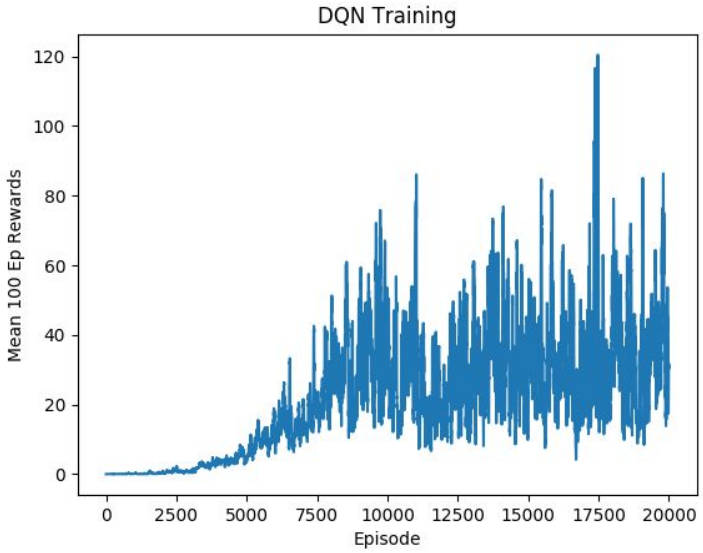
  上圖之橘線即為 PPO2 的學習曲線。最後平均分差 +9.3。
- **Compare to the vallina policy gradient**
  我們跑了多次的實驗，只更改損失函數為 PPO2 不論是在收斂的速度以及最後達到的平均分差最後都相差不多，沒有顯著的提升。

4-2

Describe your DQN model

我們使用的DQN架構為3層convolution layers再加上兩層linear layers。三層convolution的(channel, kernel_size, stride)分別為(32, 8, 4)、(64, 4, 2)、(64, 3, 1)，輸出一個64 channels的7x7圖片。兩層linear的數量分別為512、4，最後一層輸出action數量。除最後一層linear layer之外，其他的layer都使用relu。optimizer為Adam，learning rate為0.00015。
訓練過程中，make_action的eps會在episode 1~2000之間線性從1遞減至0.025，replay buffer大小為10000，每4個frame更新一次參數，每1000次update更新target_Q的參數。而在training過程中，因為每一個episode之間會更新很多次Q，因此total reward並不會是一個狀態下的結果。為了正確知道好壞，我們在每100次episode的更新後，另外test 100次episode的平均rewards。

Plot the learning curve to show the performance of your Deep Q Learning on Breakout

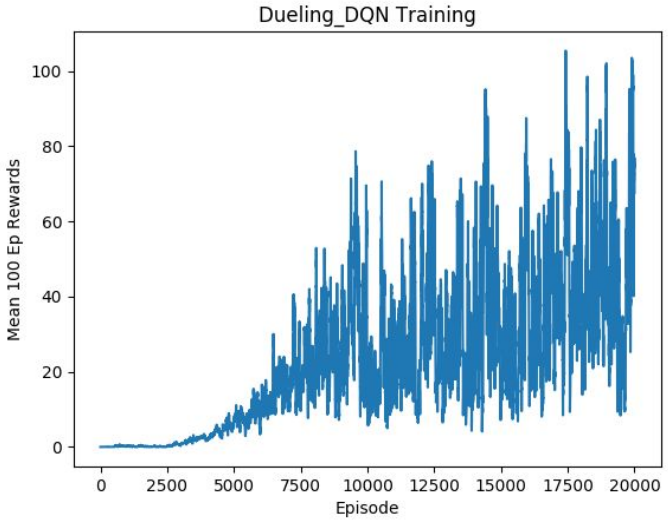


橫軸以經過第幾個episode表示，y軸為平均最後30次的rewards，可以看到大約訓練到10000個episode就已經無法更好，而且波動非常大。最終平均rewards大約落在40~60之間。

Implement 1 improvement method on page 6

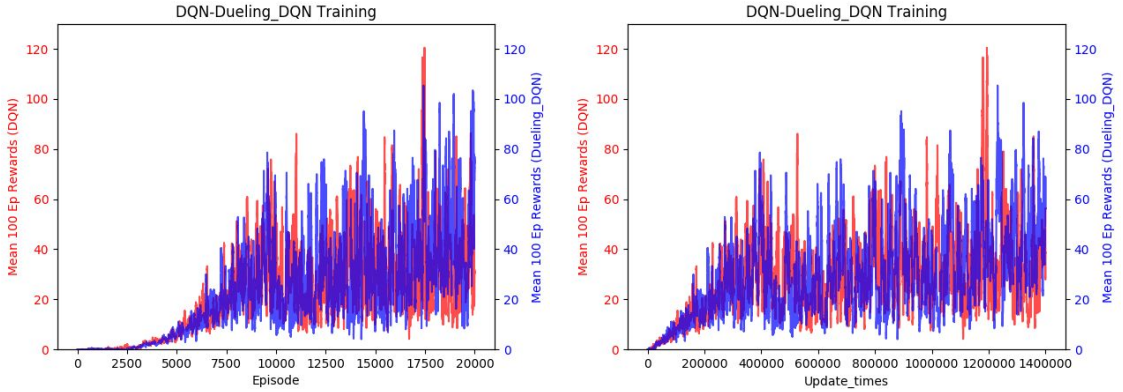
- **Describe your tips for improvement**
  我們使用的improvement為Dueling DQN，此方法只有更改DQN內部架構，其他訓練方法皆相同。一樣input會先經過同樣參數的3層convolution layers，變成一個7x7x64的output。此output會經過2次不同的2層linear layers，參數量皆和DQN相同，差別只在於一邊輸出4個action的值為adv，另一邊輸出1個值為val，最後輸出val - mean(val) + adv。

- **Learning curve**



訓練到大約10000個episode就已經無法更好，而且波動一樣非常大。最終平均rewards也大約落在60~80之間。

- **Compare to origin Deep Q Learning**

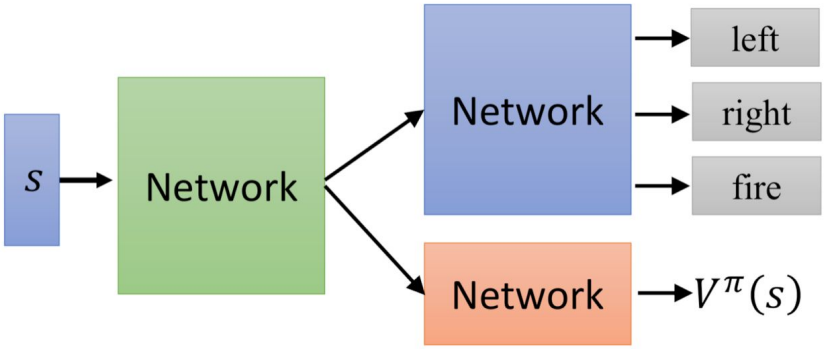


紅色的線為DQN，藍色的線為Dueling DQN，左圖x軸為episode數量，右圖x軸為update數量。從左圖以及右圖可以看出training過程是差不多的，波動也都非常大，大約到10000個episode或是400000個update過後平均reward都很難上升。就實驗結果而言，純DQN以及Dueling DQN是沒差別的。

4-3

Describe your actor-critic model on Pong and Breakout

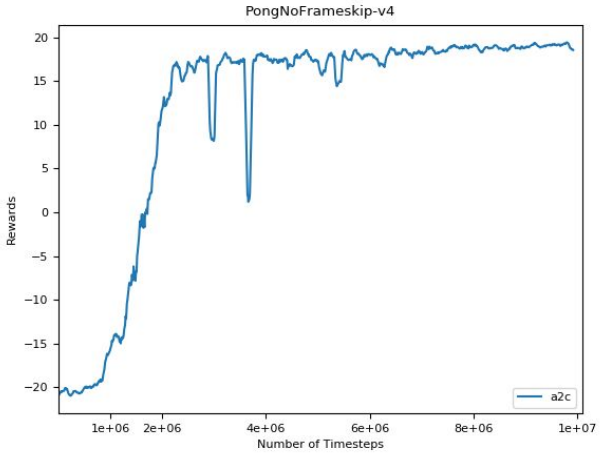
我們使用 actor-critic 的模型是 a2c ，將原本 polcy gradient 的 baseline 改為 $V^{pi}(s)$ ，並將原本的 network 多分出一層，成為以下圖示：



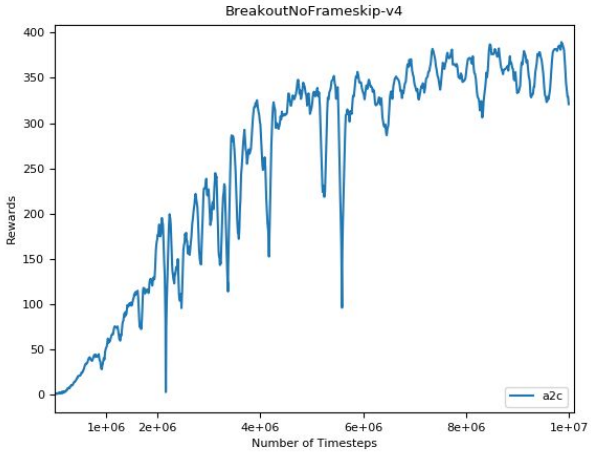
其中這個 network 的 input 為 state 之後會經過共用的 network 在分為兩個 network ，藍色的部分會 output 出做不同 action 的機率，紅色的部分便會計算出 $V^{pi}(s)$ ，我們將 a2c 這個模型分別應用在 Pong 和 Breakout 中。

Plot the learning curve and compare with 4-1 and 4-2 to show the performance of your actor-critic model on Pong & Breakout

在 4-1 中， a2c 的表現如下：



表現能夠高達將近 20 的水平比 4-1 最基本的模型高了不少，而在 4-2 部分，表現結果如下：

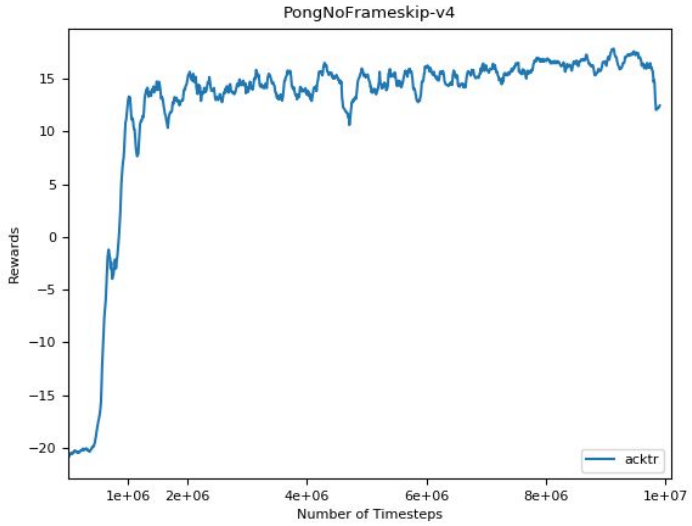


Breakout 的 a2c 表現來的比 DQN 好很多，主要原因我們認為是a2c 本來在 Breakout 中表現就會比 DQN好，再來是 update 的次數比較多，訓練也比較久，才會有這樣的結果

Reproduce 1 improvement method of actor-critic

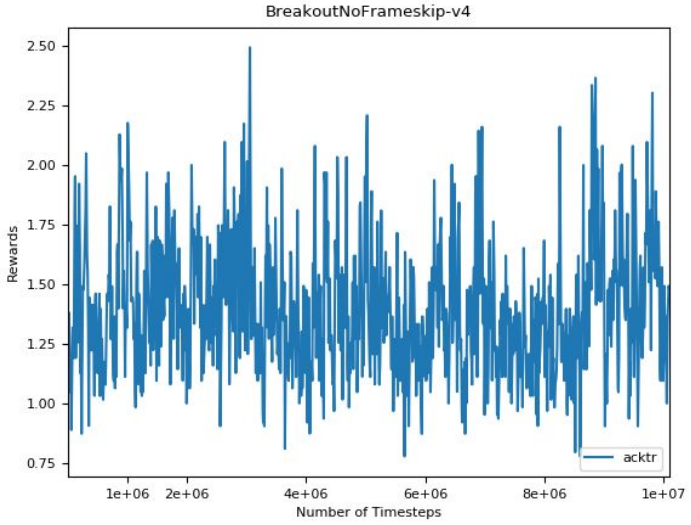
- **Describe your tips for improvement**
  在這邊我們使用的是 acktr 來精進 actor-critic 的模型，經過查閱一些文獻，他們指出 acktr 會比 a2c 來得有效率，結果也會比較好，而且訓練的時間也只會增加久一點，十分有價值，因此我們便透過 acktr 來當作比較 a2c 的模型。

- **Plot the learning curve and compare with 4-1 and 4-2, 4-3 to show the performance of your improvement**
  以下為 acktr 在 4-1 做出來的結果：



從圖中會看出 acktr 的結果似乎沒有比 a2c 來的好，但其實是很相近的，我們認為這可能只是其中一次剛好 a2c 做出來的結果比較好而已，如果多做幾次，整體表現應該會是 acktr 比較好，此外，也會發現 acktr rewards 上升的幅度比 a2c 來的早，在 1e+06 的時候就已經巨幅上升了，而 a2c 則是需要比較多 timesteps 才能巨幅上升，便如同文獻所講的 acktr 真的比較有效率。另外也會發現 acktr 做出來的曲線震盪幅度比 a2c 來的小，不會突然下降很多。

以下則為 4-2 acktr 訓練出來的結果：



在 4-2 中 acktr 的結果並不好，沒有訓練成功，經過討論我們歸納出的原因有可能是 learning rate 太大，導致 Rewards 一直在同樣的區域震幅，再來是因為我們參考別人的模型做 acktr 的，有可能是參考的那份並不是很完整，其中還有疏漏只是我們沒找出來才導致這樣的結果。

分工表	
r06725008 郭毓棠	4-3
r06725005 郝思喬	4-2
r06725020 劉冠宏	4-1