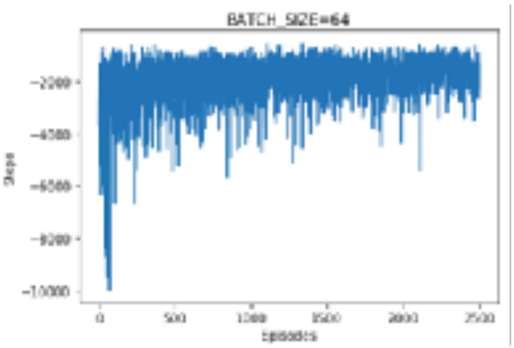
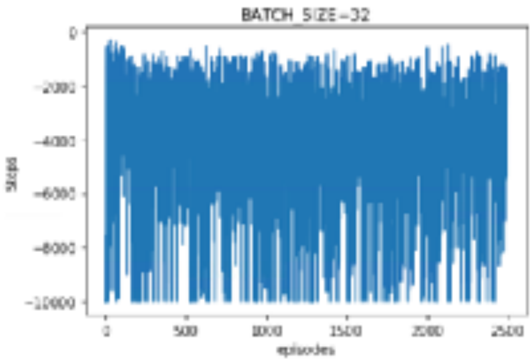


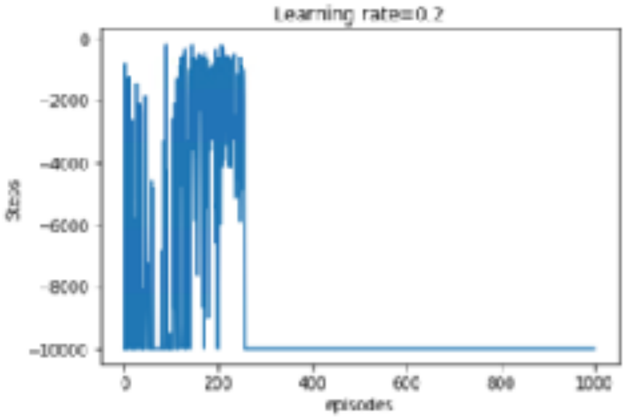
BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500



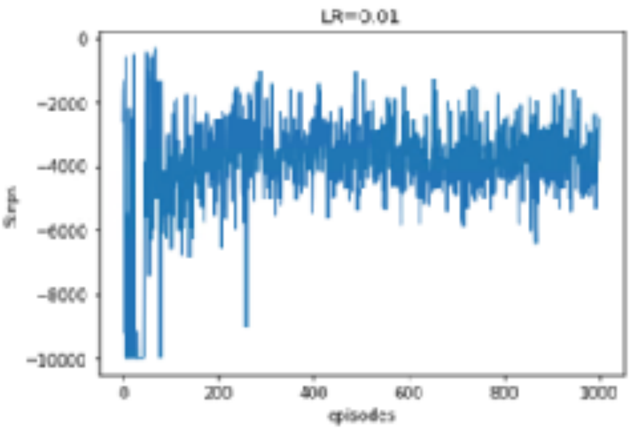
BATCH_SIZE = 32
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500

Analysis: Batch size

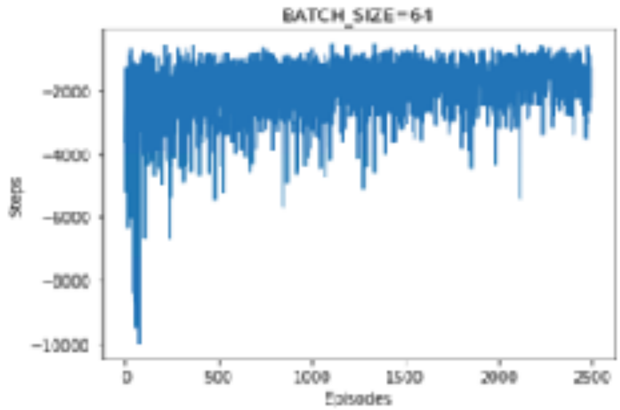
本來是將batch size設64，想說換32看看會如何，
發現episode跑2500次也不會收斂，
後來batch size也有換128，但結果也是很爛，一直收斂不了。



BATCH_SIZE = 64
LR = 0.2
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500



BATCH_SIZE = 64
LR = 0.01
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500

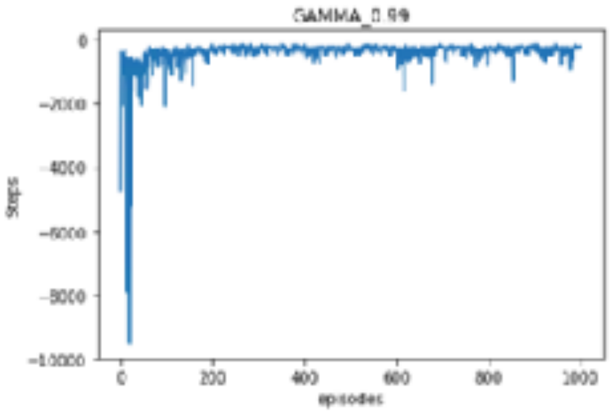


BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500

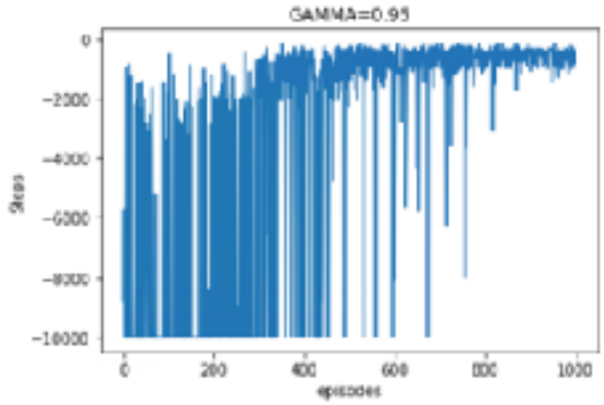
$$Q(s_t,a) \leftarrow Q(s_t,a) + \boxed{\alpha}\left[r_{t+1} + \gamma \max_{p} \boxed{Q(s_{t+1},p)} - Q(s_t,a)\right]$$

Analysis: Learning rate

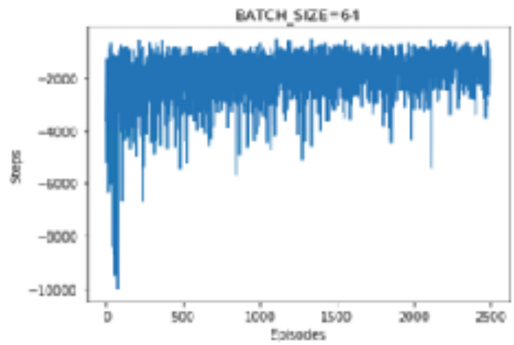
Q-Target function 中我綠色圈起來的便是我的 learning rate，可以看出設定0.2已經太大，沒收斂，0.01收斂效果也是沒有0.1好，所以0.1最合適



BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.99
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000



BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.95
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000

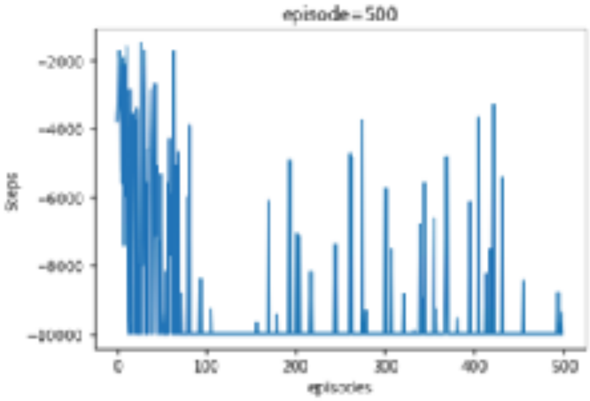


BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500

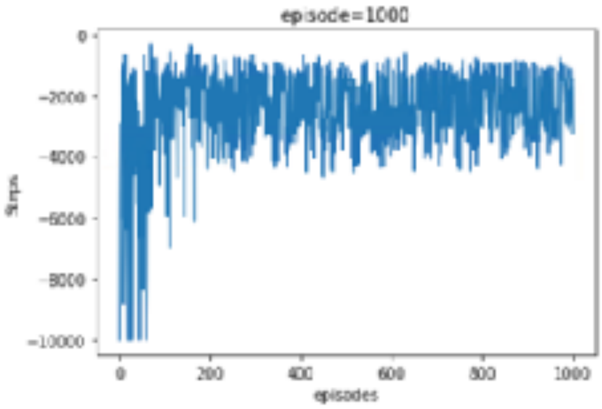
$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[r_{t+1} + \boxed{\gamma} \max_p \boxed{Q(s_{t+1}, p)} - Q(s_t, a) \right]$$

Analysis: GAMMA

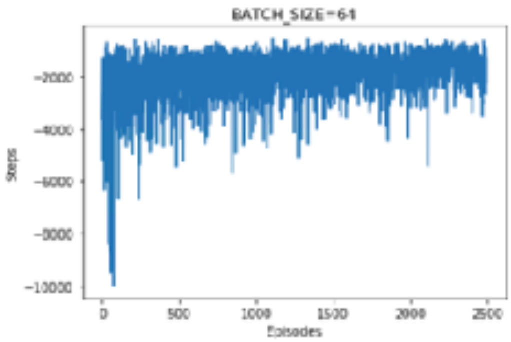
Q-Target function 中我綠色圈起來的便是我的 GAMMA，可以看出設定0.999收斂效果不錯，但 0.99的幅度就收斂的很乾淨



BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=500



BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=1000



BATCH_SIZE = 64
LR = 0.1
EPSILON = 0.1
GAMMA = 0.999
TARGET_REPLACE_ITER = 100
MEMORY_CAPACITY = 2000
episode=2500

Analysis: episode

調500太低，完全看不出什麼收斂有發生，1000就足夠看出來了

1. What kind of RL algorithms did you use? value-based, policy-based, model-based? why? (10%)

   我使用value-based 的DQN方法，Value-based，就是先评估每个action的Q值(Value)，再根據Q值求最佳的policy。
   因為覺得DQN的方法已足夠應用在MountainCar上，但如果再更高維可能就要使用其他更進階的方法了

2. This algorithms is off-policy or on-policy? why? (10%)
   on-policy:
   更新Q值時是使用既定的policy
   off-policy:
   更新Q值時是使用新的policy
   雖然DQN中的replay memory中包含2000個過去的樣本，但更新Q target function時是隨機採樣這些樣本，因此，並不一定使用當前policy的樣本，所以是off-policy

3. How does your algorithm solve the correlation problem in the same MDP? (10%)