EL2805 Reinforcement Learning
LAB 2

Rohit Kini
rrkini@kth.se

Neil Pradhan
npradhan@kth.se

1 Cartpole Problem

(a) The states will be formulated as follows :

state number	State	minimum	maximum
0	cart position	-2.4	2.4
1	cart velocity	-inf	inf
2	pole angle	-41,8	41.8
3	pole velocity at tip	-inf	inf

The action space will be as follows :

state number	Action
0	Take Left
1	Take Right

As the states are continuous and state space is large, thus the memory and computation requirements are large. We should therefore use deep RL or functional approximation methods over standard RL methods to solve the problem. The reward is equal to 1 if the pole angle is within 12.5 degrees and the absolute value of position is less than 2.4, otherwise the reward is equal to zero.

(b) The main function description in the given code is as follows :

i. Create gym environment for Cart Pole along with describing state size and action size
ii. Create object of the class DQNAgent
iii. Initialize Q- values and Test State matrix with zeros
iv. Iterate through 10k states made for getting good Q values and store them in Test State Matrix
v. Create 2 empty lists for scores and episodes
vi. For every episode keep a Boolean flag done = True, if episode is successful it is True, else False. Initially set the flag to False, and reset the environment
vii. Get an action, go to next state and save state, action, reward, next state, done into a buffer
viii. If done is True which means the action is good and saves the cart pole, save the score and maximum Q value and update the model with these good weights according to the frequency inputted by us.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

(c) The pseudo code for Deep Q learning is as follows :

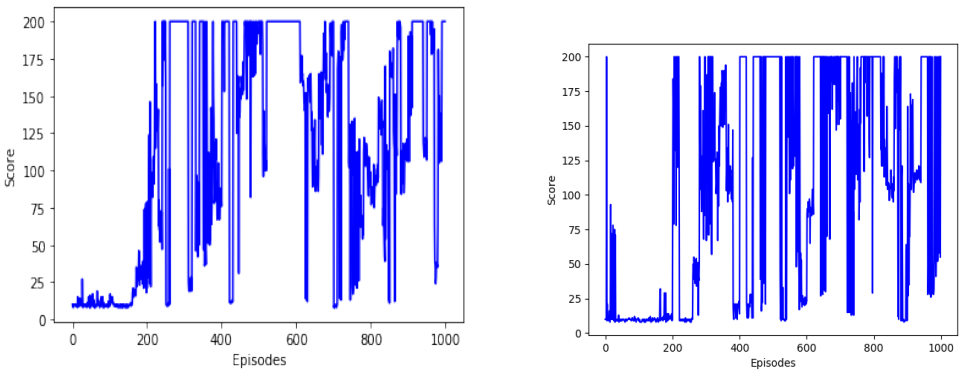
i. **Initialization:** θ and ϕ, replay buffer B, initial state S1

ii. **Iterations:** For every t >= 1,
compute π_t the ϵ-greedy policy w.r.t. Q_θ
take action a_t according to π_t, and observe r_t , s_{t+1}
store (s_t, a_t, r_t, s_{t+1}) in B
sample k experiences (s_i, a_i, r_i, s_{next_i}) from B
for i $\in$ [1; k]:
if episode finishes in s_{next_i}: $y_i = r_i$
else $y_i = r_i + \lambda$ max Q_ϕ (s_{next_i}, b)
update θ as :
$\theta \leftarrow \theta + \alpha$ (y_i - Q_θ (s_i ,a_i))$\Delta_\theta Q_\theta$ (s_i ,a_i)
after certain frequency (or steps) :
$\phi \leftarrow \theta$

(d) Neural Network Architecture
We have used simple **dense** layers with 2 and 3 layers. The loss function we have used is **MSE**, for activation function **ReLu** and **tanh** have been tried out. Finally for the optimizer we have used **Adam**.

(e) Greedy Algorithm
The following greedy algorithm is used to compute π_t.
if a = *argmax* Q(s, a):
$\pi_t = \epsilon$/m + 1 - ϵ
else:
$\pi_t = \epsilon$/m
Here m is number of possible actions.

(g) There was no significant difference between two layer and three layer neural network. The neural network of 24 x 24 and 128 x 64 gave the best results. **MSE** was best suited as loss function and for activation function **ReLu** gave the best results.

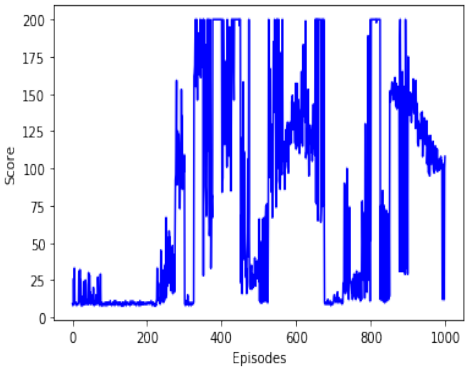


(a) Score of 128x64x2 Network at Frequency 10 (b) Score of 24x24x2 Network at Frequency 20

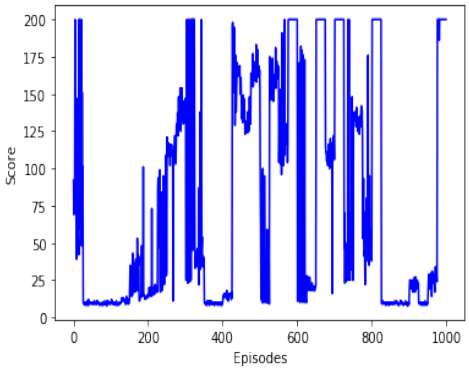
Figure 1: Few Results from two layered neural network

(h) Model Investigation: We have tested this on 24x24x2 Neural Network at frequency 25. Only one parameter is changed at a time, rest of them are kept at default value.

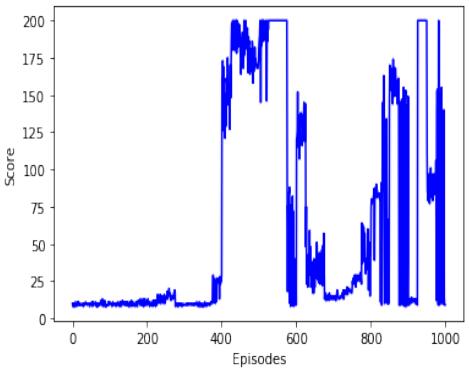
- Discount Factor



(a) Discount Factor 0.85



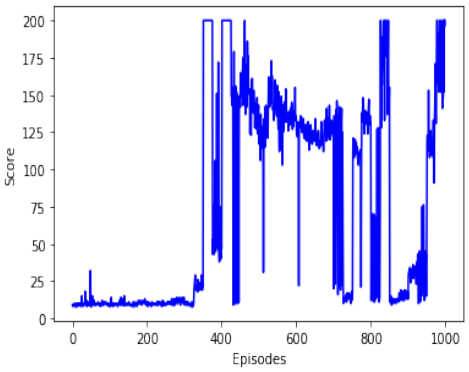
(b) Discount Factor 0.9



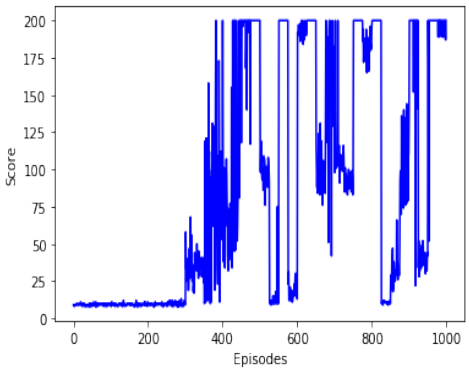
(c) Discount Factor 0.95

Figure 2: Variation in response of the network with respect to discount factor

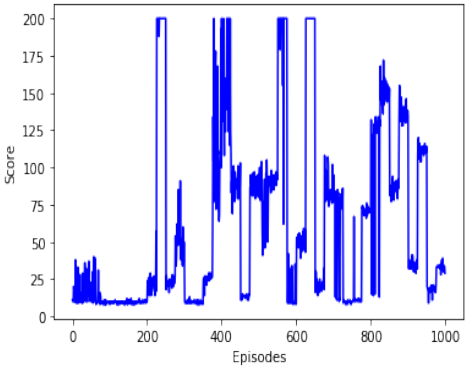
- Learning Rate



(a) Learning Rate 0.001



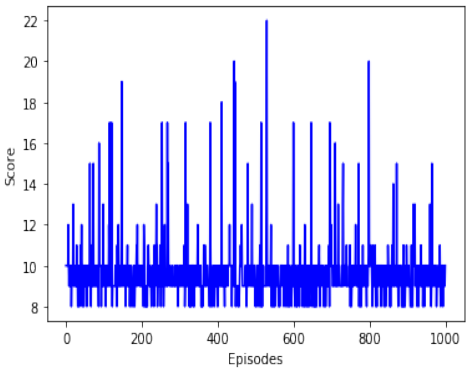
(b) Learning Rate 0.005



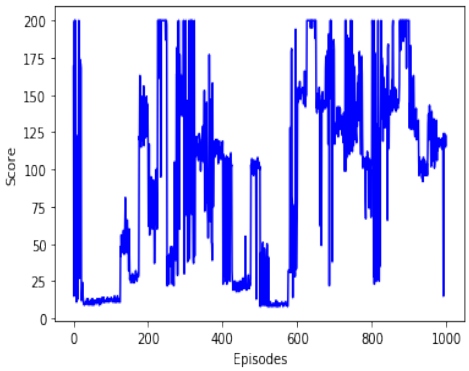
(c) Learning Rate 0.01

Figure 3: Variation in response of the network with respect to learning rate

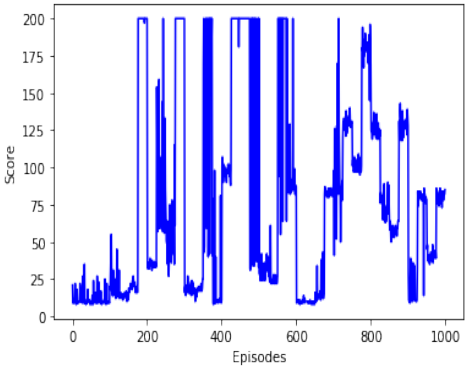
- Memory Size



(a) Memory Size 500



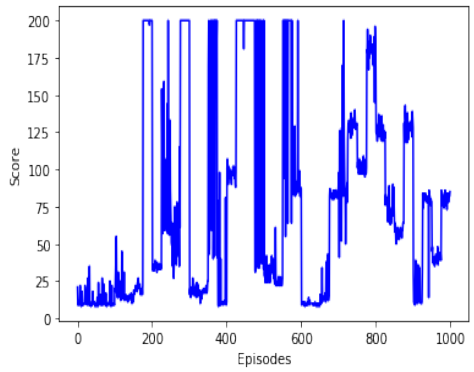
(b) Memory Size 1000



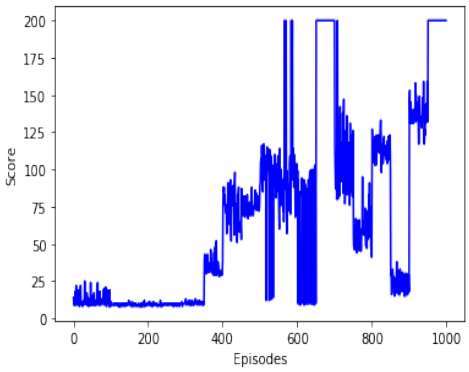
(c) Memory Size 2000

Figure 4: Variation in response of the network with respect to memory size

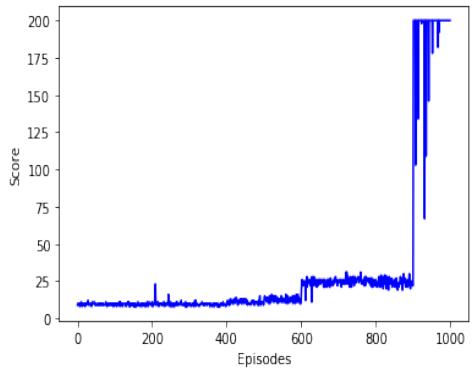
(i) Investigation of change of update of frequency.
For Network of 24x24x2. With Learning rate of 0.005, Discount rate of 0.95 and Memory size of 2000. We have computed the response of network at update frequency of 25, 50 and 100.



(a) Score at Frequency 25



(b) Score at Frequency 50

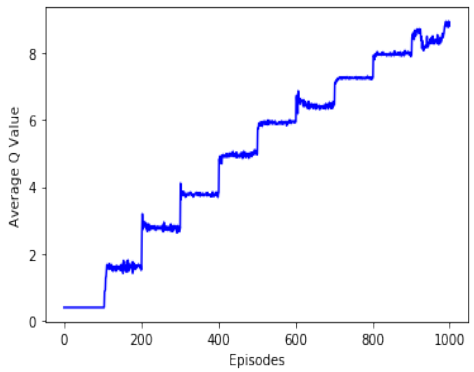


(c) Score at Frequency 100

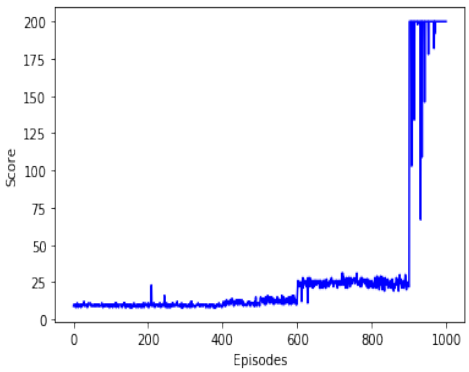
Figure 5: Analyzing response of network to change in update frequency.

We observed that for a batch of episodes the pattern remains the same, i.e. for one batch the values will be high and for upcoming batch the values will be low. This cycle gets repeated after set frequency. This can be seen in figure 5. The frequency of changes in scores is low in high frequency (100) and high in low frequency (25). That is with longer frequency the score sustains for longer time.

(j) Best solution that we got that solved the problem was of Network architecture 24x24x2 with 2000 memory size and 100 update frequency. The other hyper parameters were set at default value.



(a) Q value



(b) Score

Figure 6: Best Solution

From figure 6(b) it can be seen that the score for last 100 episodes is almost 200 with scores of few episodes falling down 200.