Parmeter Name:

S: current State, A: current Action, R: Reward, S'=next State

W^-=weights of Separate target Network

Q_Target_Next Equation And Q_Target:

Q_target_next= max$\hat{q}$(S',A,W^-)

Q_target=reward +(gamma*Q_target_next*(1-done))

My Loss function and Optimizer is

Loss function=(1/2)*(Q_Expected-Q_target)^2

Optimizer: Adam, Learning rate: 0.0005

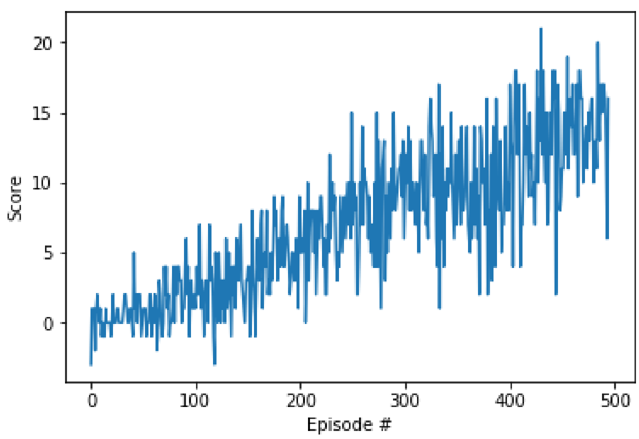
Batch_size=64:

My Qnetwork consists of this.

Input Layer vector:37 , hidden1 Layer size: 64 ,hidden2 Layer size:64, output Layer size: 4

My Plot of Rewards

```
Episode 100	Average Score: 0.97
Episode 200	Average Score: 3.52
Episode 300	Average Score: 7.54
Episode 400	Average Score: 9.50
Episode 495	Average Score: 13.03
Environment solved in 395 episodes!	Average Score: 13.03
Score: 16.0
```



DQN Algorithms

If(memory size<BASTCHSIZE)

Choose action A from state S using policy epsilon-greedy action.

Take action A, Observe reward R, input frame xt

Prepare next state: Next state s’=∅(xt,xt+1,xt+2,xt+3)

Store experience tuple(S’,A,R,S) in memory D

Else

Obtain minibatch of tuples(sj,aj,rj,sj+1) from D

Q_target_next= max$\hat{q}$(S’,A,W^-)

Q_target=reward +(gamma*Q_target_next*(1-done))

Update ∇w=alpha*(yj- $\hat{q}$(S’,A,W^-)) ∇$\hat{q}$(S’,A,W^-)

W^-=W

Performance Improve

At First, I want to try Deep layer and big size. And Dqn is weakness that is overestimated Q-value. Because choosing max q-value. If I use Dueling DQN, I will get more good performance than dqn. Because Dueling Dqn is separated value function and advantage function.