Deep RL

Here, we code up the DQN for the CartPole experiment using Tensorflow. The goal here is to get a score greater than 195 for 100 consecutive trials which denotes the task as solved.

We have used `keras` library for building the neural network containing two hidden layers with `tanh` activation function and a linear output layer.

We try varying the hyper-parameters used in constructing the DQN such as `epsilon, hidden layer size, minibatch size` to compare the variation in performance.

We have kept the following hyper-parameter values fixed:

```
replay memory size = 10000
epsilon decay = 0.995
epsilon min = 0.01
discount factor = 0.99
# hidden layers = 2
learning rate = 1e-4
```

We have tried different sets of hyper-parameters and corresponding learning curves and the observations are summarised below:

```
epsilon = 0.4, minibatch size = 32, hidden1 size = 64, hidden2 size = 32,
target update freq = 50
```

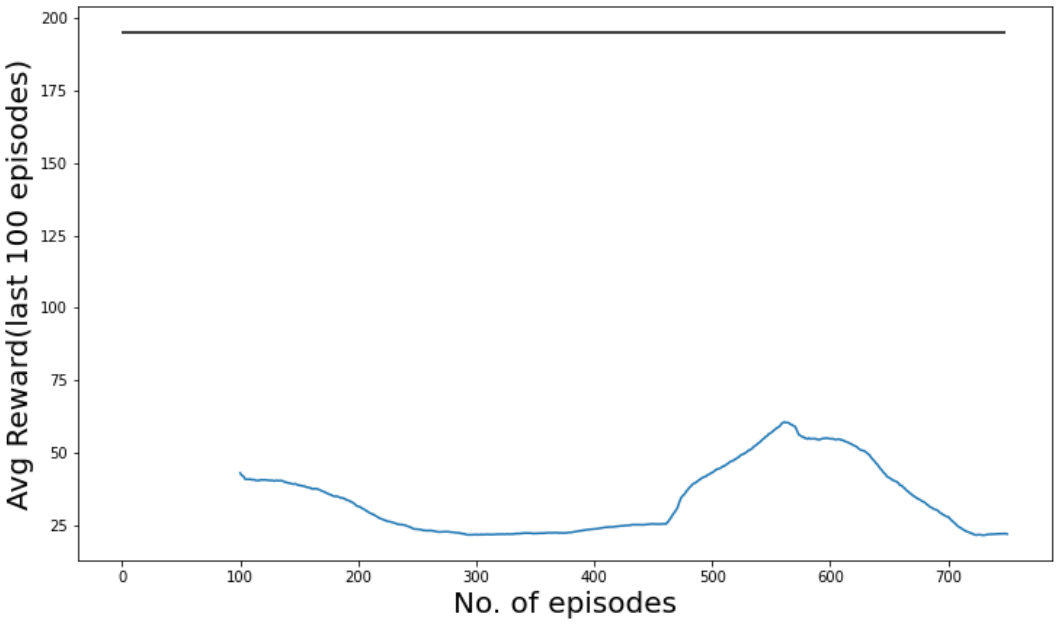

Figure 1: Learning Curve

The above parameter choice clearly gives poor results. This can be attributed to poor network construction (small hidden layer sizes). We need to make the network more non-linear, hence we try increasing the `hidden2 size` to 64.

`epsilon = 0.4, minibatch size = 32, hidden1 size = 64, hidden2 size = 64, target update freq = 50`

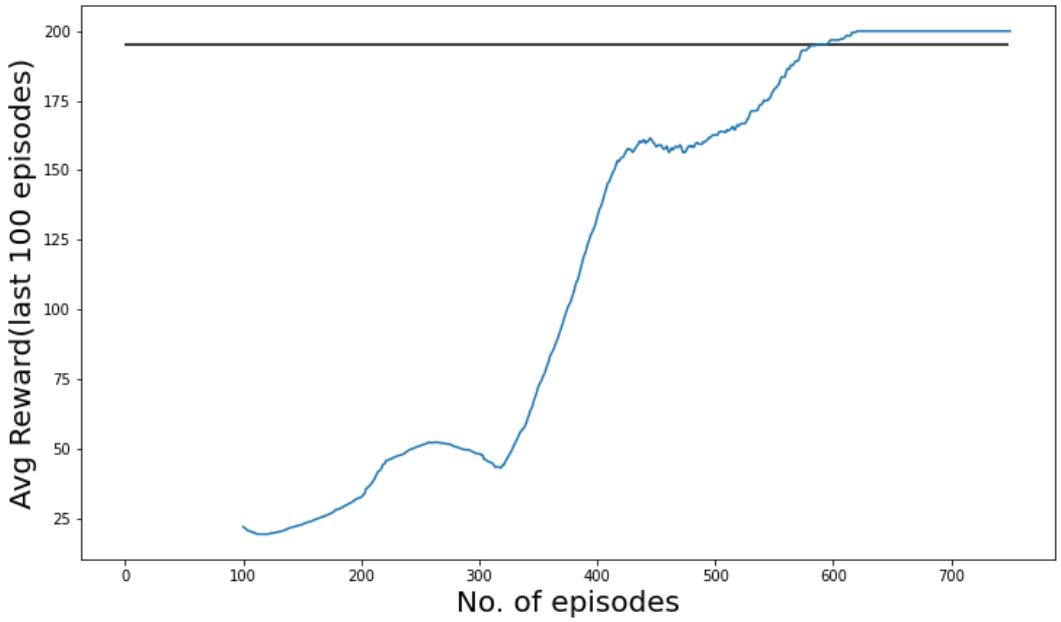

Figure 2: Learning Curve

This gives much better results than before. We observe that making the network a bit more complex works. But we can see that after some episodes the slope of the curve decreases to zero (even becomes negative) and then starts to increase again. This may be because the `target update freq` is small. Now we further increase `hidden1 size` to 128. `epsilon` is increased to 0.6 for further favouring exploration. `minibatch size` is increased to 64 to include more samples for learning and `target update frequency` is increased to 150 for decreasing the correlation between q network and target network. If the target network is very much correlated with q network, it can slow down the progress as weight updates in the network are smaller.

(Best Hyper-parameters)
`epsilon = 0.6, minibatch size = 64, hidden1 size = 128, hidden2 size = 64, target update freq = 150`

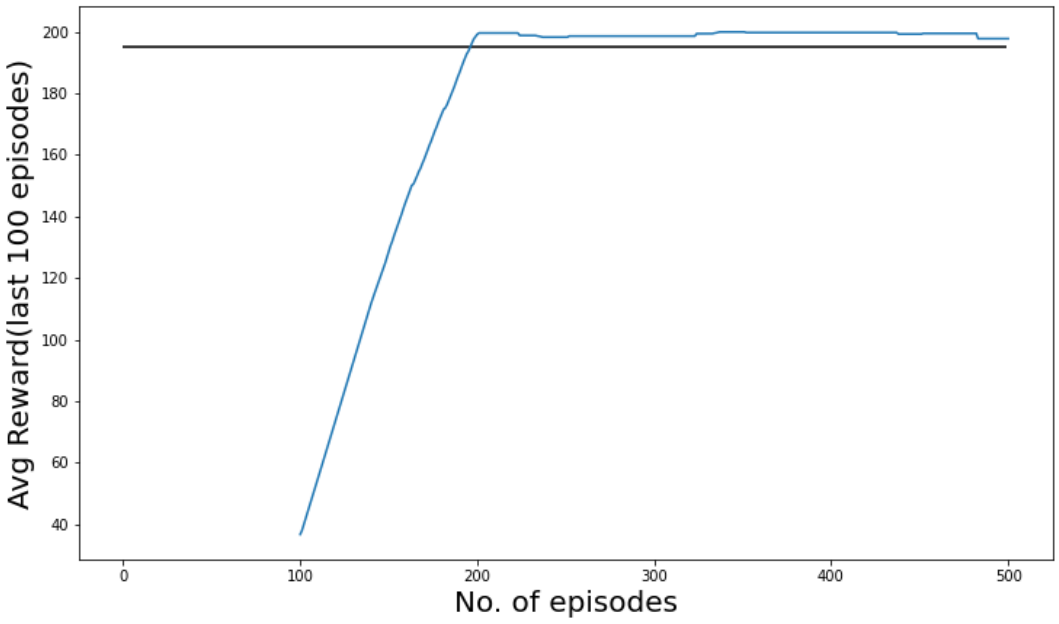

Figure 3: Learning Curve

For this choice of hyper-parameters we get the task solved much earlier in nearly 200 episodes as compared to the previous choice which took about 600 episodes. Also the problems faced in the previous iterations seems to be resolved. Hence we finalize this to be the best set of hyper-parameters as far as the task completion is concerned.

Now we try to run the DQN by removing the target network which can be done by setting `target update freq = 1` and setting other hyper-parameters to the best values found earlier.

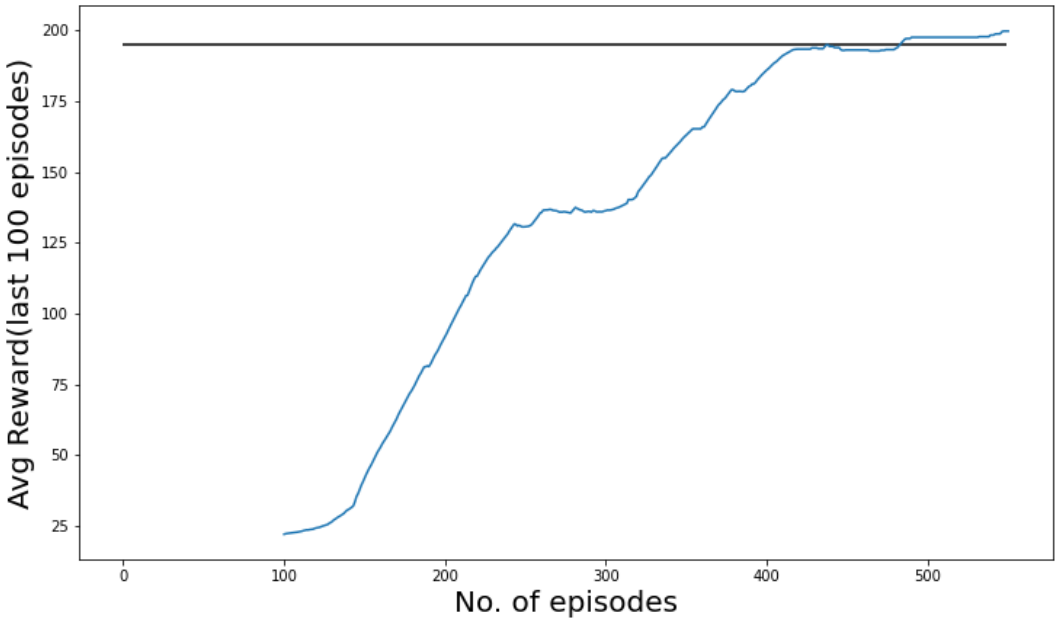

Figure 4: Learning Curve

As discussed earlier, we can see that removing the target network slows down the performance and the task completion takes more than double the episodes (but indeed solves the task) as compared to when the target network is used.

Now we remove the experience replay by setting `replay memory size = 1` and `minibatch size = 1` and setting other parameters to the best values.

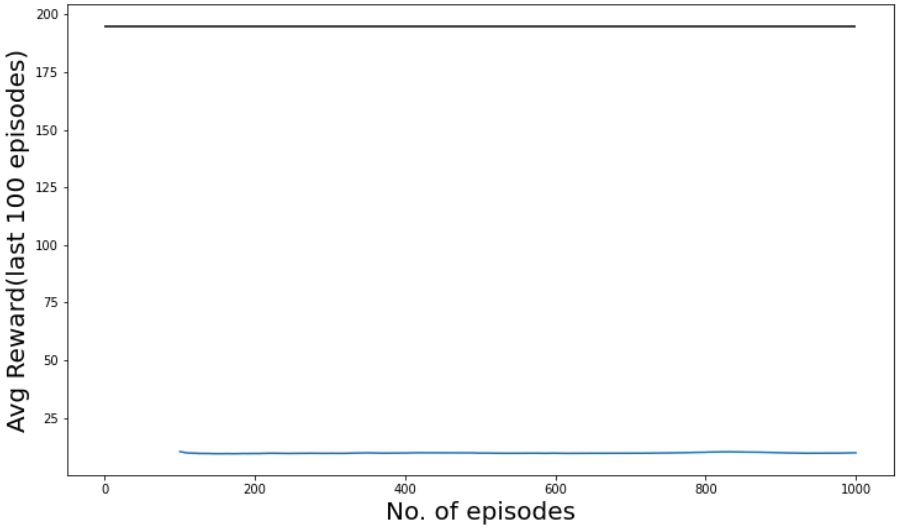

Figure 5: Learning Curve

Then, we remove both experience replay and the target network keeping other best hyper-parameters.

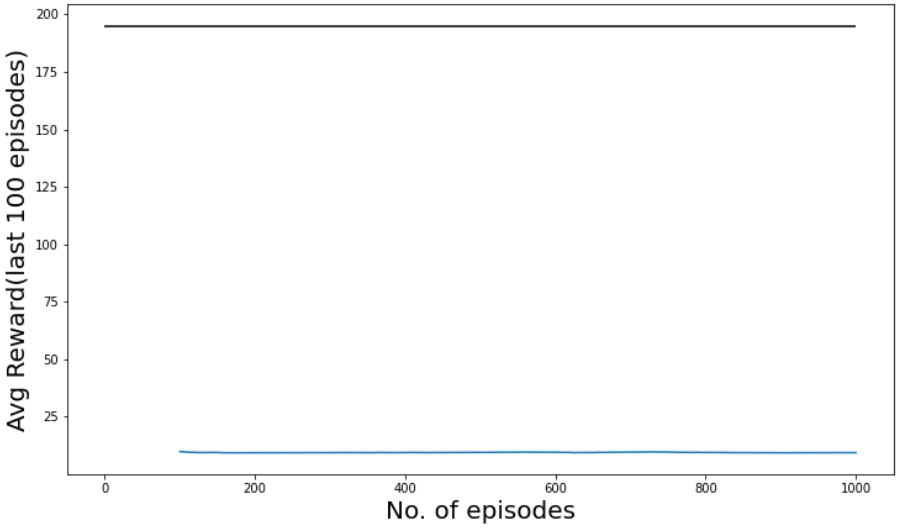

Figure 6: Learning Curve

We can clearly see that experience replay plays a significant role in improving the performance. Experience replay helps in improving the sample efficiency by reusing the stored samples for quality learning and removing it hinders the training and there is no progress and task is not completed.