Amina Tabassum

NUID:002190127

Question 01:

The plots for model trained with hidden layers of dimension 8,16,64 and 128 are shown below.

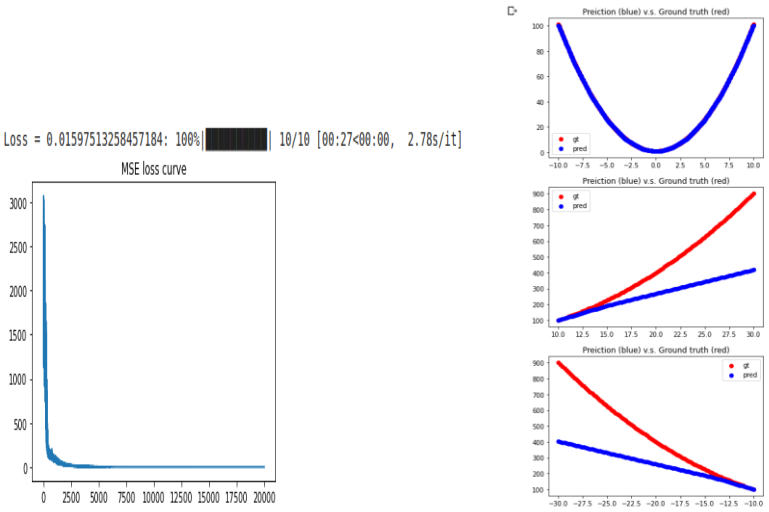

Figure 1: Hidden layer dimensions=8

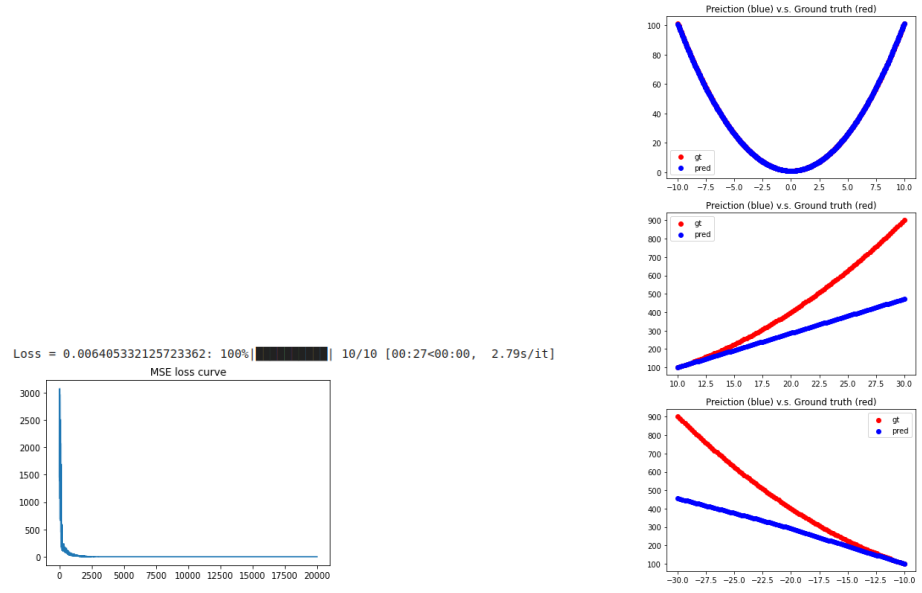

Figure 2: Hidden layer dimensions=16

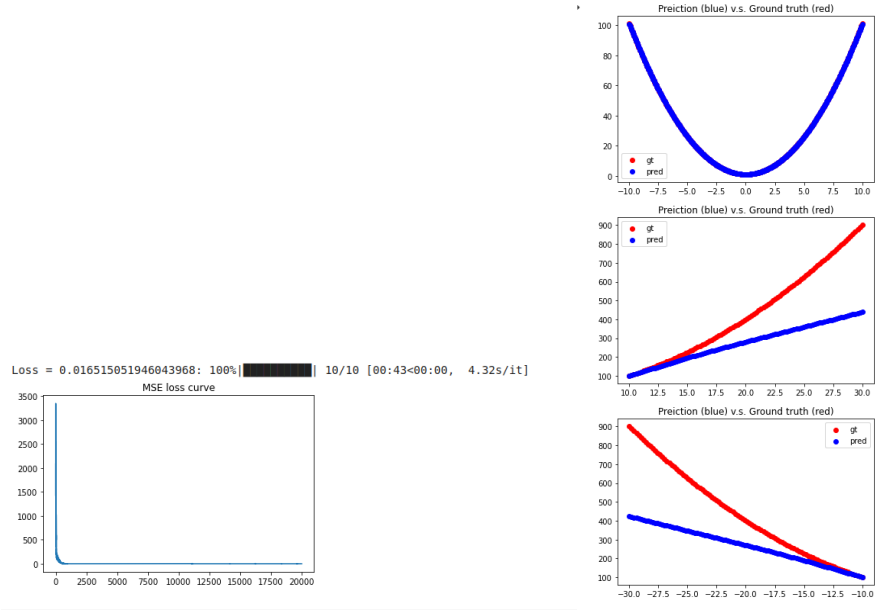

Figure 3: Hidden layer dimensions=64

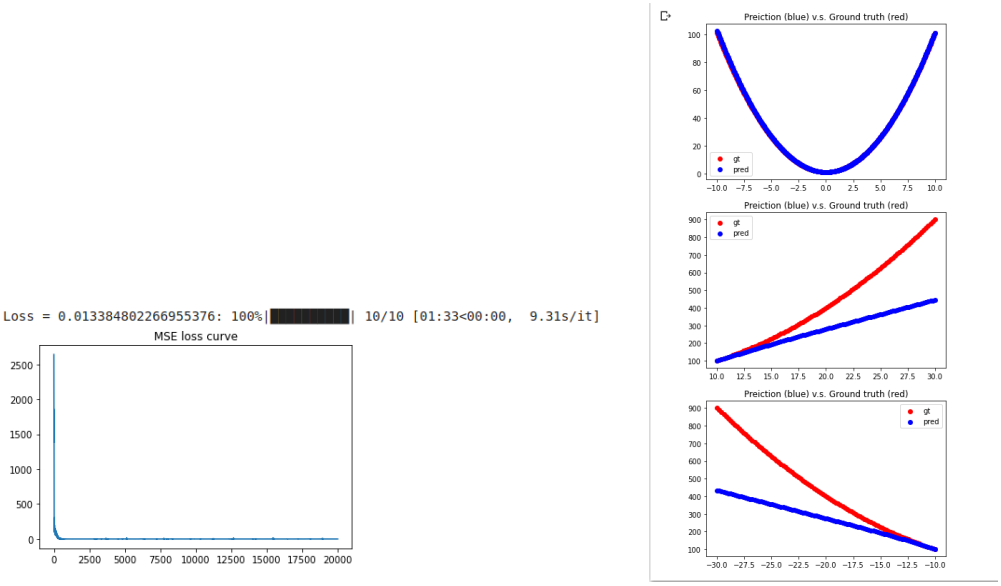

Figure 4: Hidden layer dimensions=128

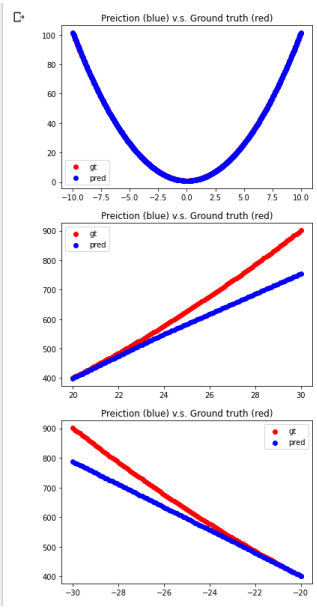

Figure 5: Learned Model within range of –20 and 20

As we increase the dimension of hidden layers, the loss decreases. Initially, for dimension=8 the loss was 0.159 and it reduced to 0.0164 for hidden layer dimension =16 and ultimately reduced to 0.0133 for hidden layer dimension=128. Moreover, the learning curve also increases and gap between predicted and ground truth values for test data decreases with increase in dimensions of layers. In other words, model learns. In deep learning, dimension and size of hidden layers is actually hyperparameter and sometimes it may help improve performance and sometimes it may just improve complexity. Here, it is helping improve accuracy of model. With reference to above figures, my ground truth and predicted model are exactly same and hence model is very accurate within –10 and 10 range. As figure 5 shows that MSE loss decreases outside the range of –10 and 10 as well. The gap between true and predicted curve is small and slope increases. Increase in slope means increase in learning rate.

Question 02:

Results of DQN and double DQN are shown below:

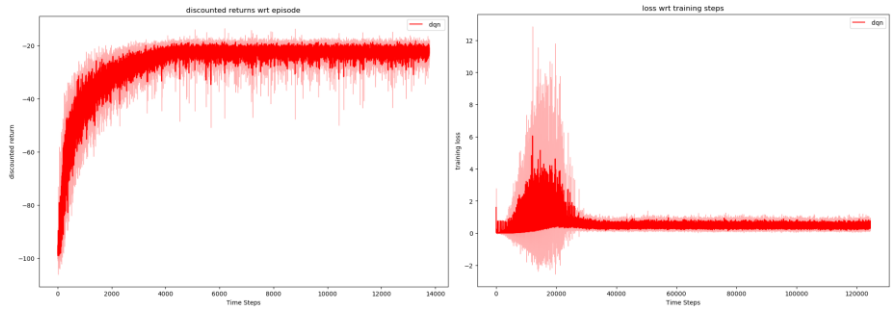

Figure6: DQN on Four rooms environment

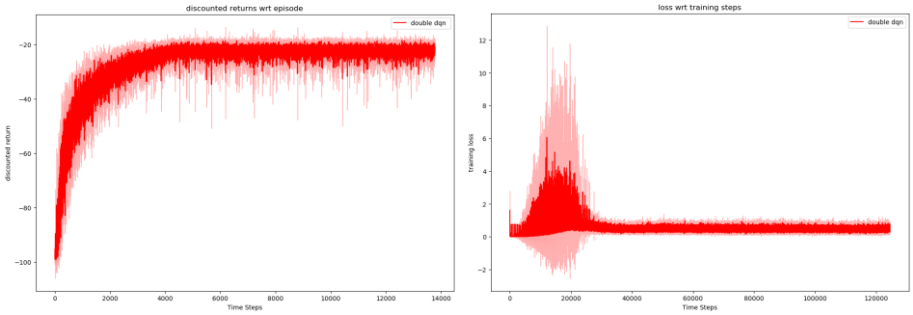

Figure7: Double DQN on four rooms environment

The DQN estimates the q value by maximizing the the q_values over all actions I.e. it always chooses greedy action during update stage. This maximization operation introduces bias which can be observed in figure 6. Since, agent always selects actions with highest q values and hence the bias can be observed around 2000 timesteps where agent settles relatively faster as compared to double DQN. In double DQN, we use different estimators for action selection and different estimator function for q value update. Other than this, variance is high in both plots.

Question 03:

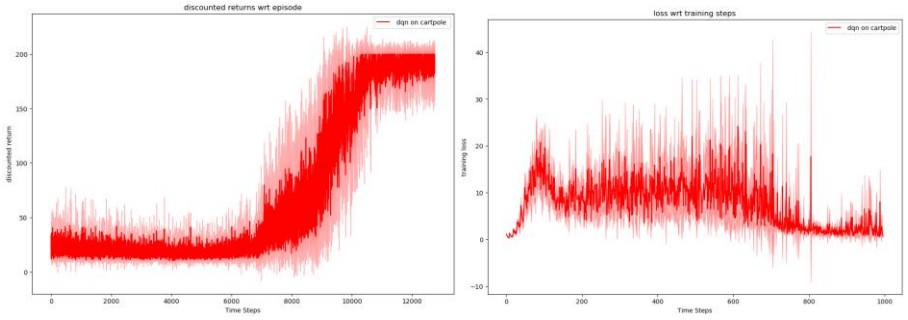

Figure8: DQN on cartpole environment

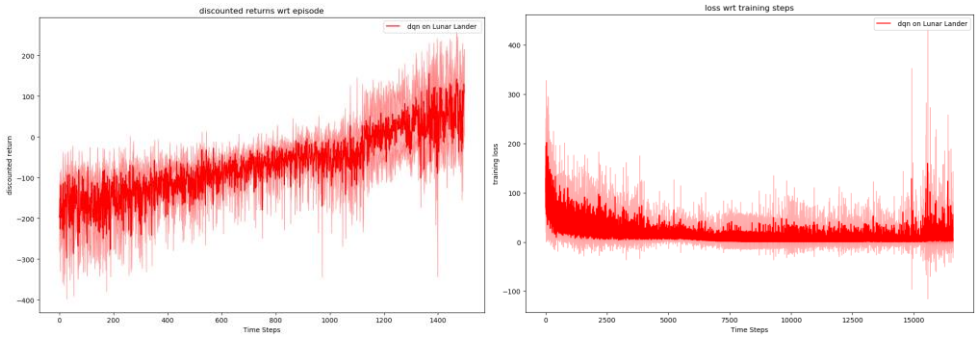

Figure9: DQN on Lunar Lander environment

Cartpole Hyperparameters:

Observation dimensions=4

Actions dimension=2

Number of hidden layers=2

Hidden Layer dimensions=128

Gamma=0.9999

Maximum time steps per episode=200

Total running time steps=100000

Epsilon start value =1

Epsilon end value =0.01

Epsilon durartion=500,000

Replay buffer size=500000

Start training step=2000

Frequency update of behavior policy=1000

Frequency update of target policy=10000

Batch size=128

Learning rate=0.001

Lunar Lander Hyperparameters:

Observation dimensions=8

Actions dimension=4

Number of hidden layers=2

Hidden Layer dimensions=128

Gamma=0.995

Maximum time steps per episode=1000

Total running time steps=500000

Epsilon start value =1

Epsilon end value =0.01

Epsilon duration=250,000

Replay buffer size=500000

Start training step=2000

Frequency update of behavior policy=25

Frequency update of target policy=5000

Batch size=128

Learning rate=0.001

d) Extra Credit

I rendered runs at five different stages of training I.e., after every 25% of training phase. I rendered both cartpole and Lunar Lander environments and observations are as following:

For Cartpole case: When training just started (0%), the pole goes more than 15 degrees and episode ends since agent does not know what action to take. When model was 25% trained, the agent does not let the pole go more than 15 degrees by moving in right direction. Hence, it gets higher return and episode is relatively longer than 0% case. But, the agent runs to the right side and never stops. After 50% training, the agent moves in right direction but stops after 2.4 units and hence higher return and greater episode length. Since,it stops after moving 2.5 units , episode ends again and pole falls more than 15 degrees. When it has accomplished 75% training, the agent learns that only stopping after moving 2.5

units right side is not enough. It should take small steps to avoid pole falling . So, when it moves more than 2.5 units right side, it takes small steps left and right and hence stops the pole falling more than 15 degrees. But this is for very short amount of time. Sometimes, it tries to move left from very beginning which causes episode to terminate. When, it is 100% trained, it tries to avoid the right side by just running very fast to left side. However, it moves very fast to left side and pole falls and episode terminates. The agent has not achieved optimality and still needs to learn how much it should move left and right.

For Lunar Lander case, my observations from rendering are as follows:

When training starts, the lander just crashes. No thrusters or random thrusters that don't yield anything. After completing 25% training, the lander stops itself from crashing but never lands. After training half time steps, the lander does land but if it goes slightly outside the landing area whilst it is in the air, it seems to get lost and does nothing except staying afloat. (I even observed it once reaching the ground but outside the target range and hence it didn't have any idea of what action to take to reach the target area. After completing 75% training, the does not get lost if it goes slightly outside the target area. Infact, it finds a way back into it. However, if it receives too much thrust either right or left, it moves very far from target and again gets lost. After 100% training, the agent does not apply too much thrust either left or right in the beginning and hence does not get lost. However, it does get lost at the end of episode and hence it still needs to achieve optimality.

Question 04 : Atari DQN

These are the best results I could achieve. I could not find time to fine tune parameters since it was taking very long to run it.