SNIPPET 1

```
    ### START CODE HERE ### (≈ 3 lines of code)
    model.add(Dense(128, input_dim = state_dim, activation = 'relu'))
    model.add(Dense(128, activation = 'relu'))
    model.add(Dense(action_dim, activation = 'linear'))

    ### END CODE HERE ###
```

One of the functions of neural networks is to predict Q(S, A) given input vectors of S and A, equal to the target R + Y*max Q(S', A'). If the neural network is good at predicting Q(S,A) for different states (S) and Actions (A) then we have a good approximation of the Q-function. A 2 hidden layer neural network is giving the best values of reward, so we have implemented the same with 'relu' activation function for the hidden units and 'linear' activation function for the output layer. The hyperparameter tuning is discussed in more detail later in the report.

SNIPPET 2

```
### START CODE HERE ### (≈ 1 line of code)
self.epsilon = self.min_epsilon + ((self.max_epsilon - self.min_epsilon) * math.exp(-self.lamb * self.steps))
### END CODE HERE ###
```

We have implemented the exponential decay formula for finding the Epsilon. The role of epsilon is to determine the explorating rate vs exploitation rate. As we play more and more episodes, epsilon will get smaller and smaller, thereby decreasing the number of random moves.

SNIPPET 3

```
        ### START CODE HERE ### (≈ 4 line of code)
        if (st_next == no_state).all():
            t[act] = rew
        else:
            t[act] = rew + self.gamma * np.amax(q_vals_next[i])

    ### END CODE HERE ###
```

In Q-learning, we choose the Action based on the highest expected future reward. 'Q-function' is used to calculate this. This is a math function that takes two arguments: the current state, and a given action.

While in state S, we estimate the future reward for each possible action A. We assume that after we have taken action A and moved to the next state S', everything works out perfectly.

The expected future reward Q(S,A) for a given a state S and action A is calculated as the immediate reward R, plus the expected future reward thereafter Q(S',A'). We assume the next action A' is optimal.

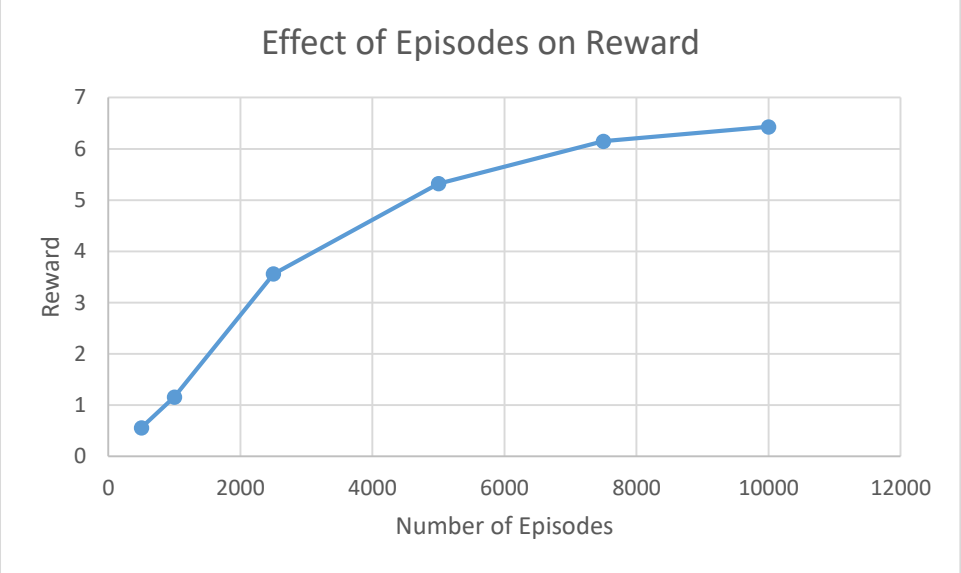
Because there is uncertainty about the future, we discount Q(S',A') by the factor gamma γ.

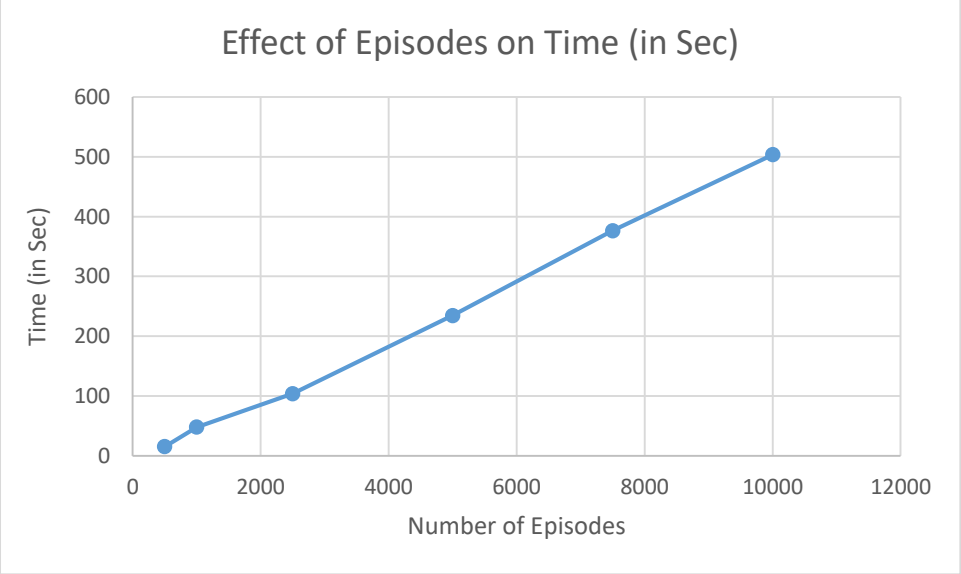
$Q(S,A) = R + \gamma * \max Q(S',A')$

EFFECTS OF VARIOUS HYPER-PARAMETERS

Effect of Number of Episodes:

As the number of Episodes increase the Reward also increases as the Agent will be better able to learn about the environment. To achieve optimum value, we are setting the episodes = 10000



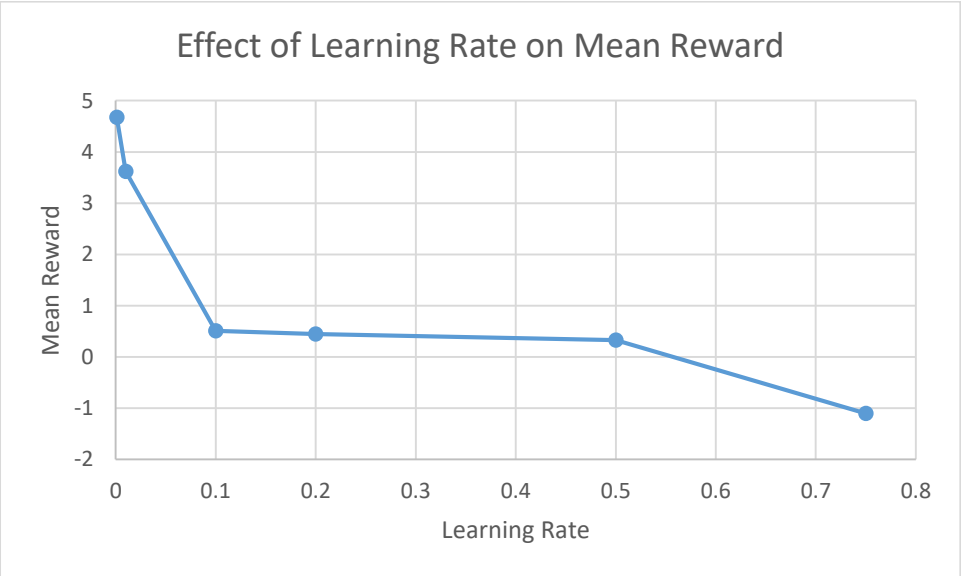


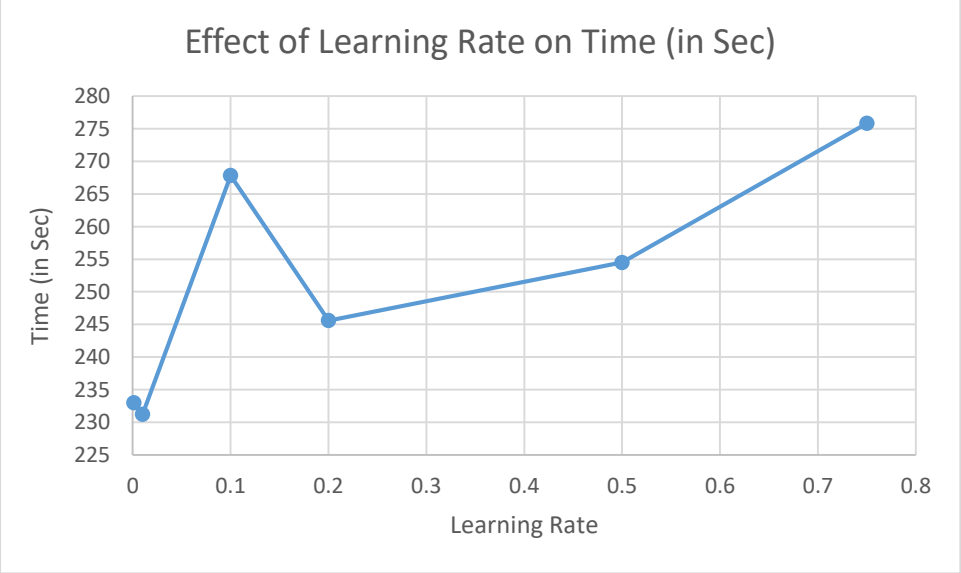
RAVI TEJA SUNKARA (50292191)

Episodes	Time (in Sec)	Reward
500	15.45	0.56
1000	47.89	1.16
2500	103.86	3.56
5000	234.43	5.32
7500	376.45	6.15
10000	503.5	6.43

Effect of Learning Rate:

As the learning rate is increasing the Mean Reward value is decreasing. So, the default value is set to very close to zero (0.00025)

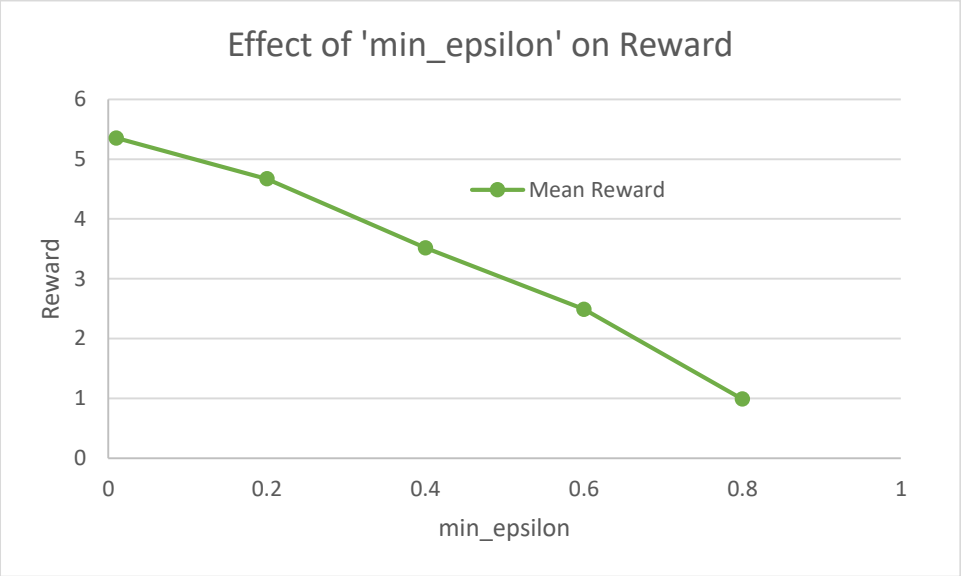


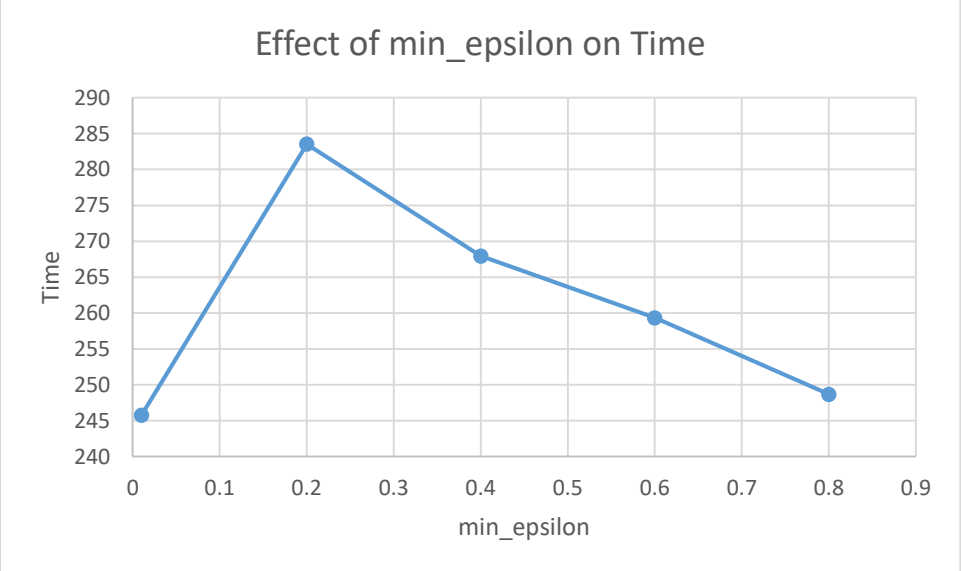


Learning Rate	Time (in Sec)	Mean Reward
0.001	232.99	4.68
0.01	231.25	3.62
0.1	267.83	0.51
0.2	245.58	0.45
0.5	254.49	0.33
0.75	275.81	-1.1

Effect of Minimum Epsilon:

As the min_epsilon value is increased, the exploration of the agent increases and as a result the reward decreases as a result of this. A min_epsilon of 0.01 is giving the best reward and also is taking the least time, so we choose this as our final parameter value.



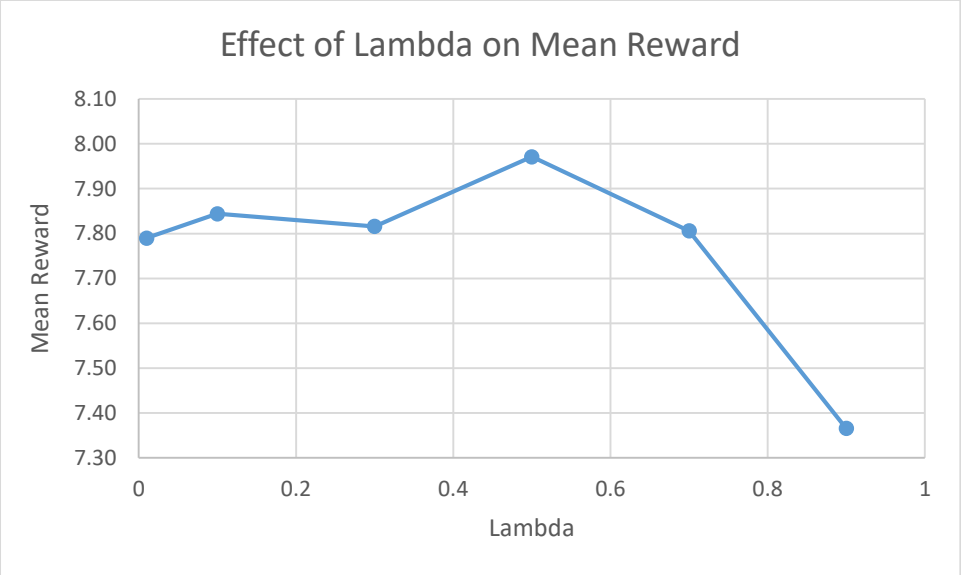


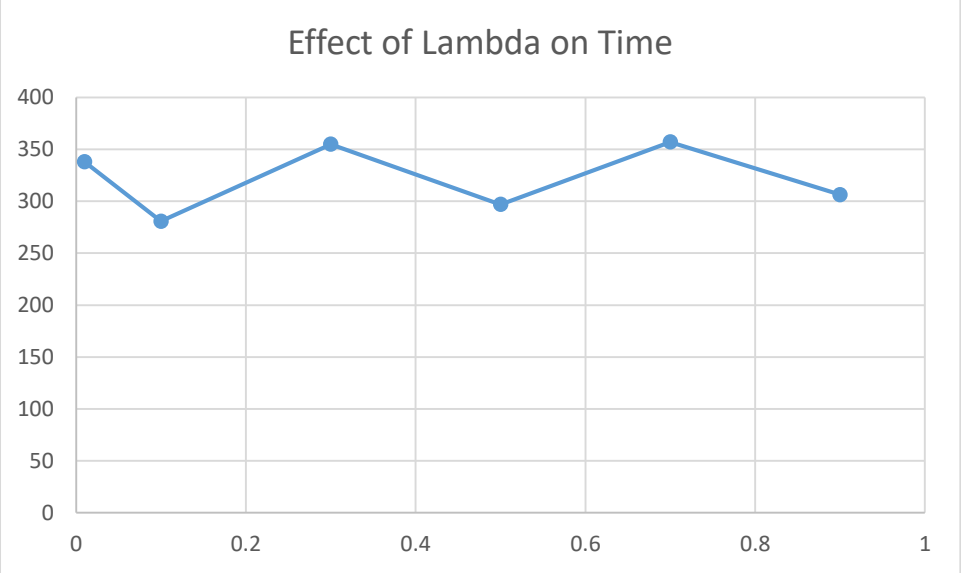
RAVI TEJA SUNKARA (50292191)

Epsilon	Time	Mean Reward
0.01	245.76	5.352
0.2	283.54	4.671
0.4	267.95	3.515
0.6	259.34	2.489
0.8	248.68	0.987

Effect of Lambda:

Lambda effects the rate of decay of epsilon. In general as lambda is increasing the reward is low.

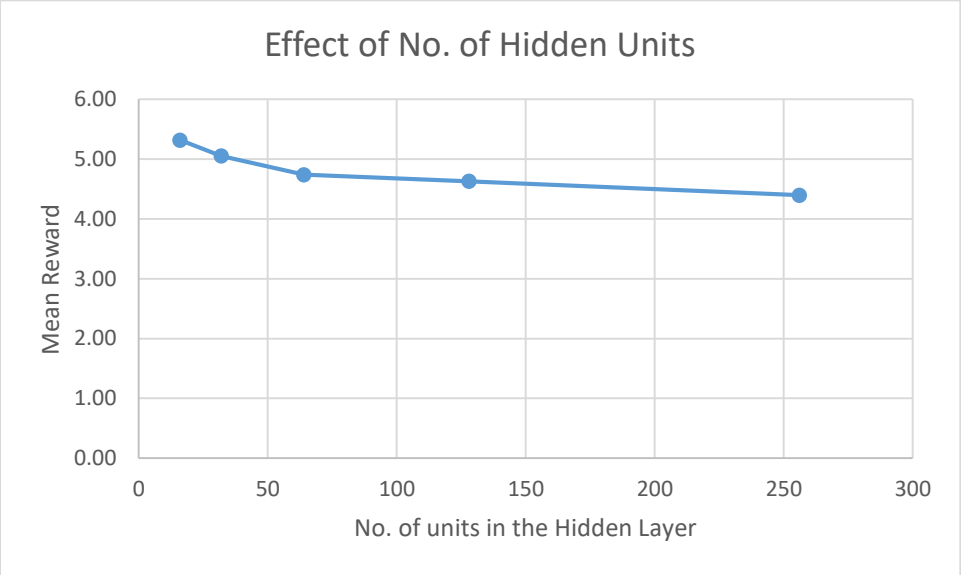


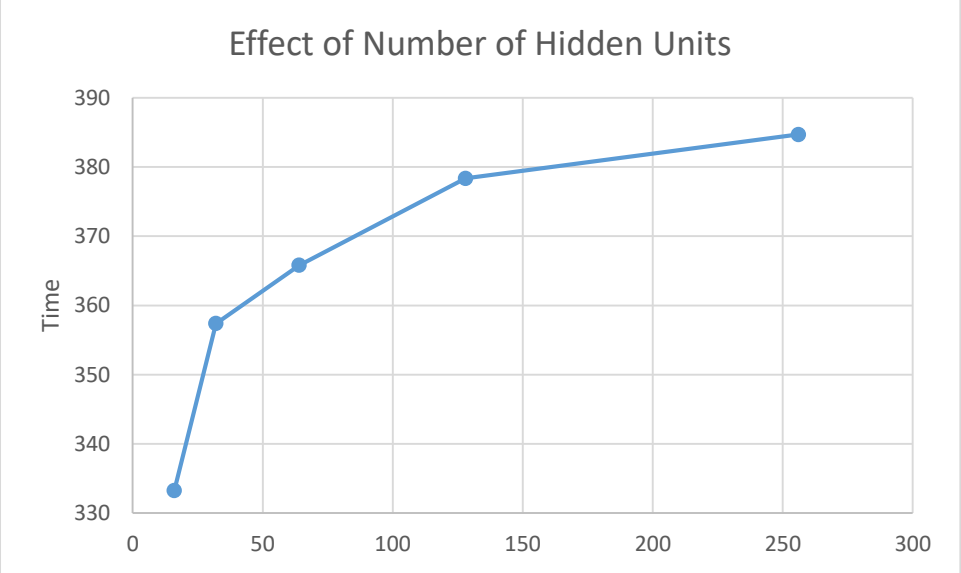


Lambda	Time (in Sec)	Mean Reward
0.01	337.83	7.79
0.1	280.75	7.84
0.3	354.85	7.82
0.5	296.95	7.97
0.7	357.08	7.81
0.9	306.29	7.37

Effect of Number of Hidden Units:

As the number of hidden units are increasing the Mean Reward value is decreasing. And as the number of hidden units is increasing, the time elapsed is also increasing because it takes more time to predict a value.



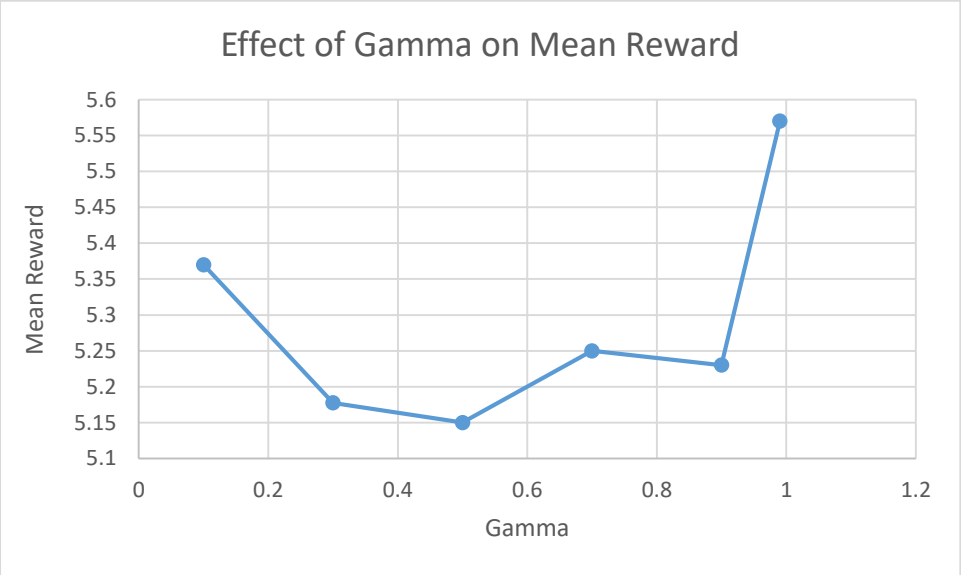


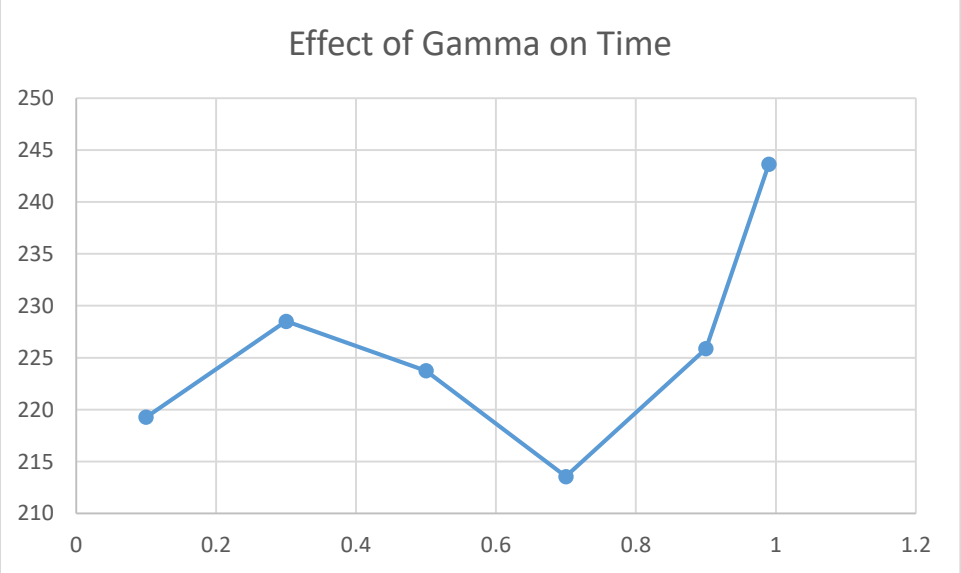
RAVI TEJA SUNKARA (50292191)

Hidden Units	Time (in Sec)	Mean Reward
16	333.26	5.32
32	357.37	5.05
64	365.79	4.74
128	378.36	4.63
256	384.69	4.40

Effect of Gamma:

High values of gamma have high mean rewards. So we use 0.99 as the default setting for the discounting factor.



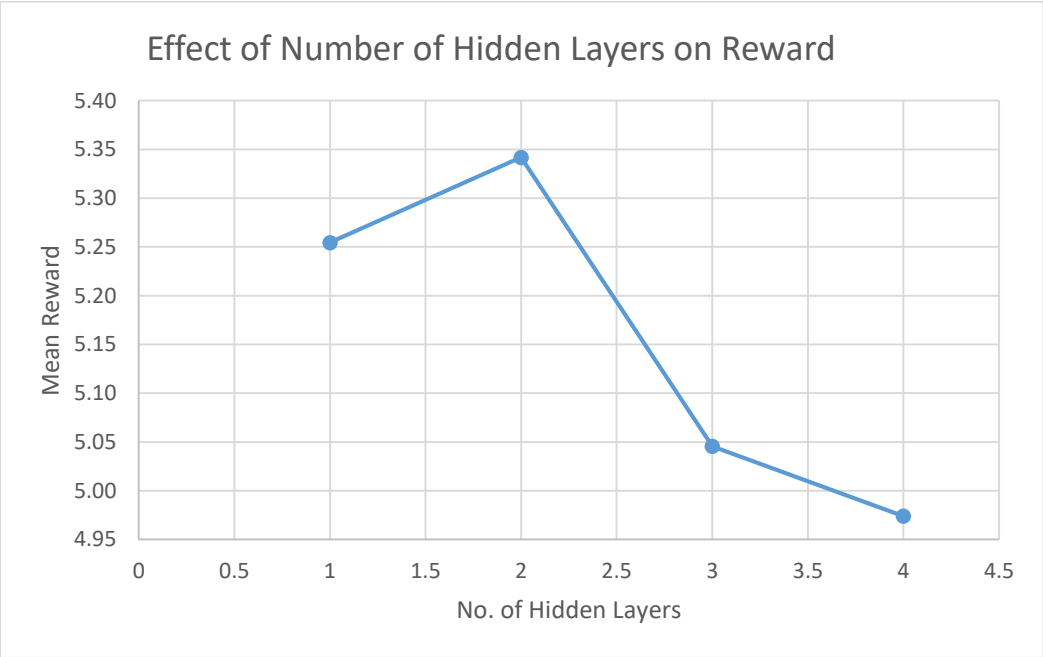


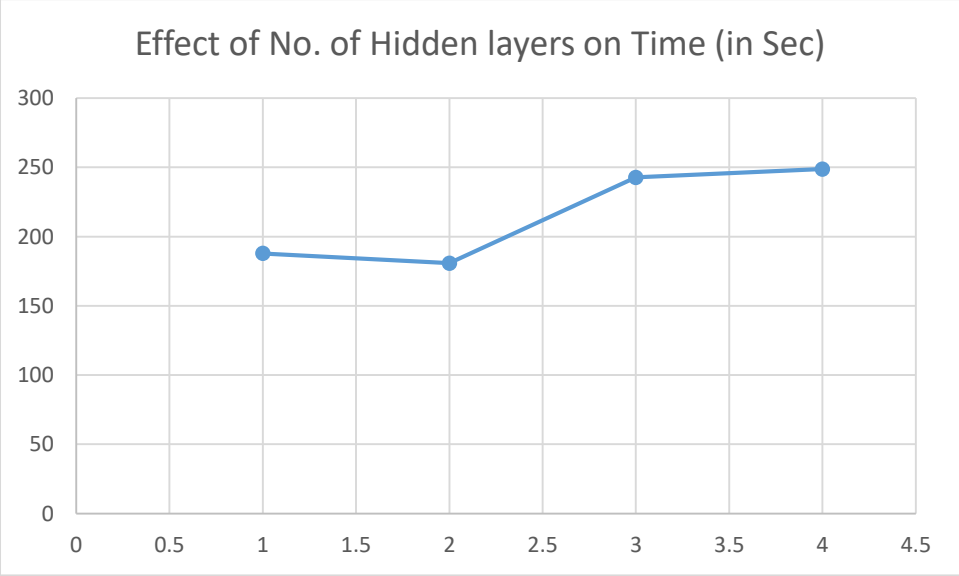
RAVI TEJA SUNKARA (50292191)

Gamma	Time	Mean Reward
0.1	219.27	5.37
0.3	228.51	5.18
0.5	223.75	5.15
0.7	213.55	5.25
0.9	225.87	5.23
0.99	243.62	5.57

Effect of Number of Hidden Layers:

For 2 hidden layers, the reward is the maximum so we have implemented 2 hidden layer neural network.



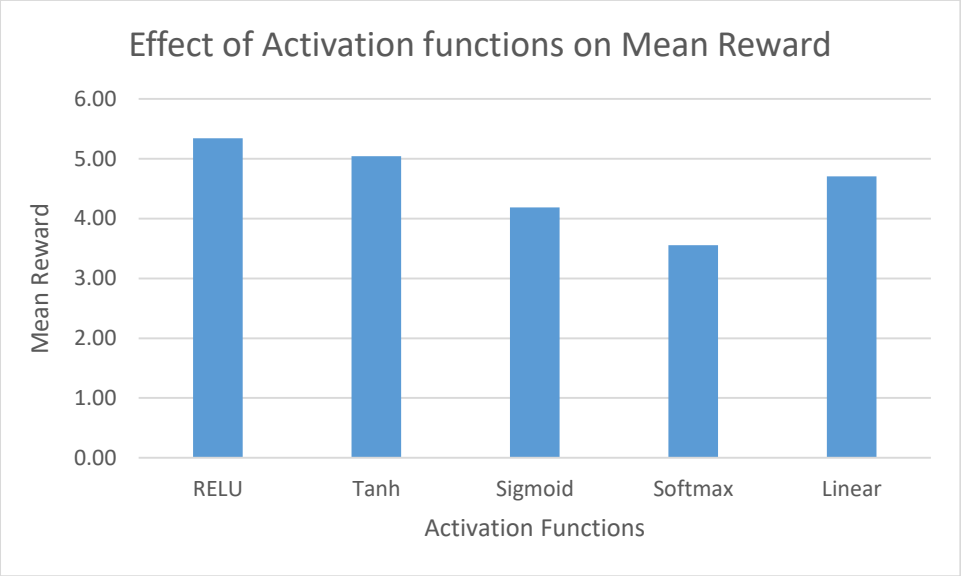


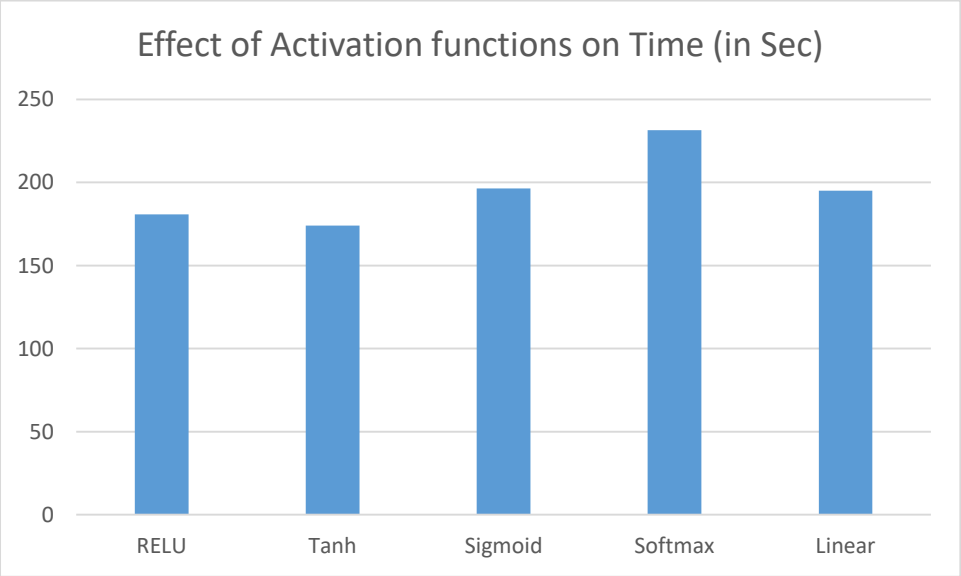
RAVI TEJA SUNKARA (50292191)

No. of Hidden Layers	Time (in Sec)	Mean Reward
1	187.78	5.25
2	180.79	5.34
3	242.76	5.05
4	248.73	4.97

Effect of Activation Functions:

Using RELU activation function we were able to get maximum reward in least amount of time.





Activation	Time (in Sec)	Mean Reward
RELU	180.79	5.34
Tanh	173.98	5.04
Sigmoid	196.4	4.19
Softmax	231.5	3.56
Linear	194.99	4.71

WRITING TASKS

Question 1:

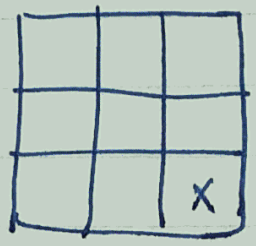
It gets stuck in non-optimal policies because it does not explore enough to find the best action from each state.

To explore, it can pick random actions occasionally. An alternative is: initialize the Q-function to values that encourage exploration. If the Q-values are initialized to high values, the unexplored areas will look good, so that a greedy search will tend to explore.

Question 2:

Initial states

State 4

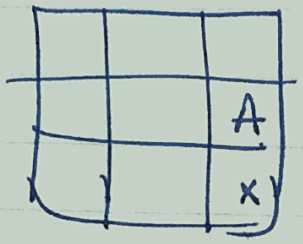


S_4

Q values for S_4 will be 0

$U = R = L = 0$

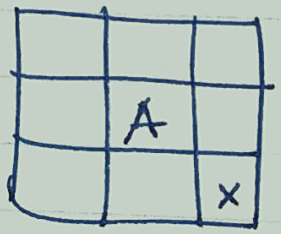
State 3



$S_{3U} = -1$ $S_{3R} = 0$

$S_{3D} = +1$ $S_{3L} = -1$

State 2

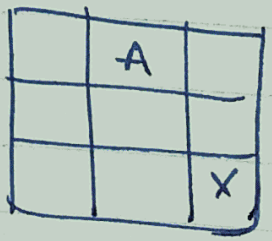


$S_{2U} = -1$ $S_{2R} = +1 + 0.99$

$= 1.99$

$S_{2D} = +1$ $S_{2L} = -1$

State 1

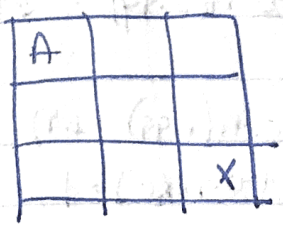


$S_{1U} = 0$ $S_{1R} = +1$

$S_{1D} = +1 + 0.99(1.99)$ $S_{1L} = -1$

$= 2.97$

state 0



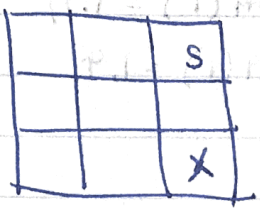
$S_{0U} = 0$ $S_{0R} = 1 + 0.99(2.97) = 3.94$

$S_{0D} = +1$ $S_{0L} = 0$

Updated States

Updated S3:

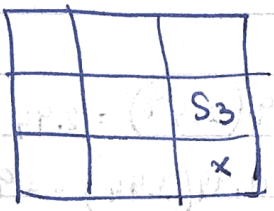
S_{31}



$S_U = 0$; $S_D = 1 + 0.99 = 1.99$

$S_L = -1$; $S_R = 0$

S_{32}



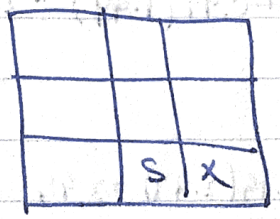
$S_{3U} = -1 + 1.99(0.99) = 0.97$

$S_{3D} = 1$

$S_{3L} = -1 + 0.99(1.99) = 0.97$

$S_{3R} = 0 + 0.99(1) = 0.99$

S_{21}



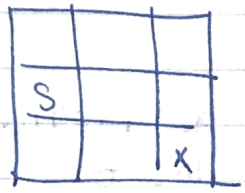
$S_U = -1 + 0.99(1.99) = 0.97$

$S_D = 0$

$S_R = 1$

$S_L = -1$

S_{22}



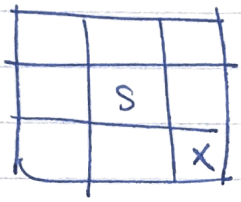
$S_U = -1 + 0.99(3.94) = 2.90$

$S_L = 0$

$S_R = 1 + 0.99(1.99) = 2.97$

$S_D = 1 + 0.99(0) = 1$

S_{23}



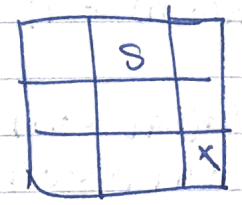
$S_U = -1 + 0.99(2.97) = 1.94$

$S_L = -1 + 0.99(2.97) = 1.94$

$S_D = +1 + 0.99(1) = 1.99$

$S_R = 1 + 0.99(1) = 1.99$

Updated state of S1



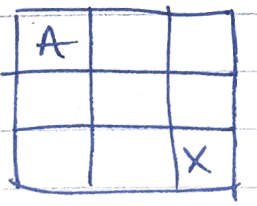
$S_{1U} = 0.99(2.97) = 2.94$

$S_{1L} = -1 + 0.99(3.94) = 2.90$

$S_{1R} = 1 + 0.99(1.99) = 2.97$

$S_{1D} = 1 + 0.99(1.99) = 2.97$

Updated state of S0

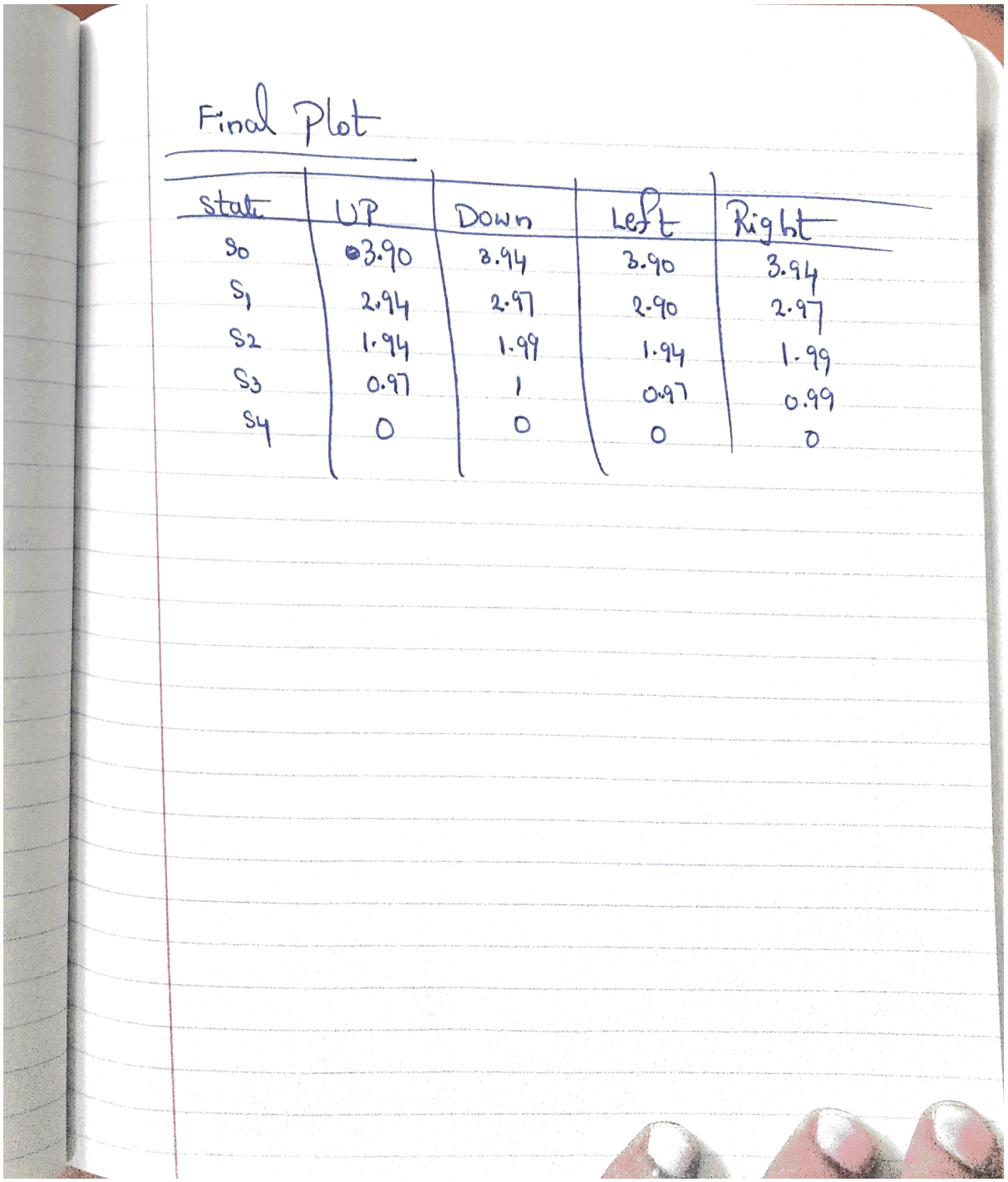


$S_{0R} = 1 + 0.99(2.97) = 3.94$

$S_{0L} = 0 + 0.99(3.94) = 3.90$

$S_{0U} = 0 + 0.99(3.94) = 3.90$

$S_{0D} = 1 + 0.99(2.97) = 3.94$



Final Plot

State	UP	Down	Left	Right
S_0	3.90	3.94	3.90	3.94
S_1	2.94	2.97	2.90	2.97
S_2	1.94	1.99	1.94	1.99
S_3	0.97	1	0.97	0.99
S_4	0	0	0	0