

## Q2

### Model Structure

My model consists of 4 linear layers that first increase in size and then decrease in size to the output size. Relu activation functions were used in the intermediary layers.

```
class DQN(nn.Module):  
    """  
    build your DQN model:  
    given the state, output the possiblity of actions  
    """  
    def __init__(self, in_dim, out_dim):  
        """  
        in_dim: dimension of states  
        out_dim: dimension of actions  
        """  
        super(DQN, self).__init__()  
        # build your model here  
        self.fc1 = nn.Linear(in_dim, 64)  
        self.fc2 = nn.Linear(64, 256)  
        self.fc3 = nn.Linear(256, 64)  
        self.fc4 = nn.Linear(64, out_dim)  
  
    def forward(self, x):  
        # forward pass  
        x = F.relu(self.fc1(x))  
        x = F.relu(self.fc2(x))  
        x = F.relu(self.fc3(x))  
        return self.fc4(x)
```

### Hyperparameters

The hyperparameters used for the DQN model were as follows

Batch size = 128

Gamma = 0.999

Eps start = 0.9

Eps end = 0.05

Eps decay = 2000

Target update = 10

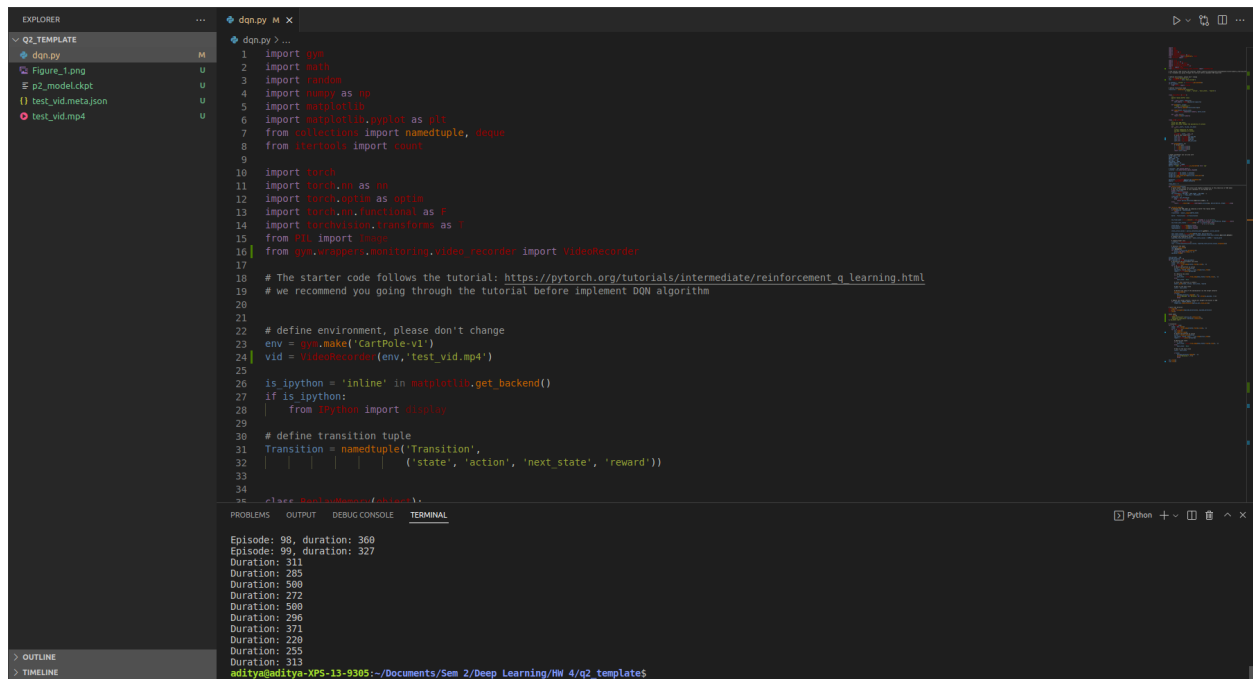
Memory capacity = 10000

Number of train episodes = 100

Optimizer = Adam

## Performance on test episodes

The model performed well on the test episodes and the mean duration of the 10 episodes was 332.3



```
1 import gym
2 import math
3 import random
4 import numpy as np
5 import matplotlib
6 import matplotlib.pyplot as plt
7 from collections import namedtuple, deque
8 from itertools import count
9
10 import torch
11 import torch.nn as nn
12 import torch.optim as optim
13 import torch.nn.functional as F
14 import torchvision.transforms as T
15 from PIL import Image
16 from gym.wrappers.monitoring.video_recorder import VideoRecorder
17
18 # The starter code follows the tutorial: https://pytorch.org/tutorials/intermediate/reinforcement\_q\_learning.html
19 # we recommend you going through the tutorial before implement DQN algorithm
20
21
22 # define environment, please don't change
23 env = gym.make('CartPole-v1')
24 vid = VideoRecorder(env, 'test_vid.mp4')
25
26 is_ipython = 'inline' in matplotlib.get_backend()
27 if is_ipython:
28     from ipython import display
29
30 # define transition tuple
31 Transition = namedtuple('Transition',
32                        ('state', 'action', 'next_state', 'reward'))
33
34
35 class DQN(nn.Module):
36     def __init__(self, n_acts):
37         super(DQN, self).__init__()
38         self.fc1 = nn.Linear(4, 16)
39         self.fc2 = nn.Linear(16, 16)
40         self.fc3 = nn.Linear(16, n_acts)
41
42     def forward(self, state):
43         x = torch.tensor(state).float()
44         x = F.relu(self.fc1(x))
45         x = F.relu(self.fc2(x))
46         x = self.fc3(x)
47         return x
48
49 def train():
50     # training loop
51     pass
52
53 def test():
54     # testing loop
55     pass
56
57 if __name__ == '__main__':
58     train()
59     test()
```

Episode: 98, duration: 360  
Episode: 99, duration: 327  
Duration: 311  
Duration: 285  
Duration: 500  
Duration: 272  
Duration: 500  
Duration: 296  
Duration: 371  
Duration: 220  
Duration: 255  
Duration: 313

aditya@aditya-XPS-13-9305:~/Documents/Sem 2/Deep Learning/NN 4/q2 template\$

## Training Process

