Report

Description of Learning Algorithm

DQN Model (Neural Network)

The learning for the Banana Navigation environment is implemented using a Deep Q Learning model. The neural network for the DQN model has two hidden layers (fully connected), each of which have 64 neurons. The input space vector has 37 dimensions that pertain to the agent's velocity and perception of objects around the agent's forward direction. The output vector of the model relates to the four possible movement actions the agent can take. While the research paper uses a CNN for image representation, the Banana navigation environment is represented using the input vector state. The hyperparameters used in the model are shown in dqn_agent.py.

The parameters used for the DQN model are as below.

- Maximum number of episodes =2000
- Maximum number of time steps per episode= 1000
- Starting value of epsilon= 1.0
- Ending value of epsilon= 0.01
- Decay rate for epsilon = 0.995

DQN Algorithm for Banana Navigation

The following are the steps involved in executing the DQN model for the Banana Navigation environment.

1. Create an object (agent) of class Agent and initialize the state size, action size and the random seed.
2. Set the values of the parameters as indicated above.
3. Initialize an empty list (Scores) to keep the scores from each episode.
4. For each state, take an action, get the values of the next state and rewards, and append the list (Score).
5. Update the value of epsilon for the next episode based on the decay rate.
6. Compute the mean of the reward for every 100 episodes.
7. Terminate the algorithm when the average value of the reward for 100 episodes equals 13.0
8. Chart the average values of the reward for every 100 episodes to understand the progression of the average cumulative reward.

Results and Discussion for Future Work

For the given set of parameters outlined in the report, the DQN model has converged in less than 500 episodes (482 episodes for this training). While this is encouraging, there are improvements that can be realized by changing hyperparameters such as the number of layers and neurons in the layers; and the decay rate for epsilon. Further, the improvements suggested in the last section of the DQN lesion (Double Deep Q Networks, Prioritized Experience Replay, and Dueling Deep Q Networks) can also enhance model performance.

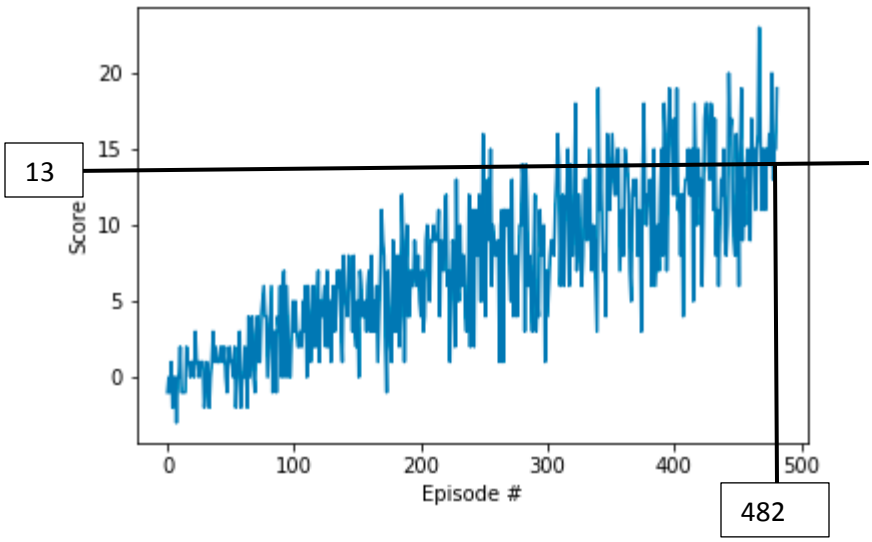

Figure: Scores (Reward) vs. Episodes for the DQN Model