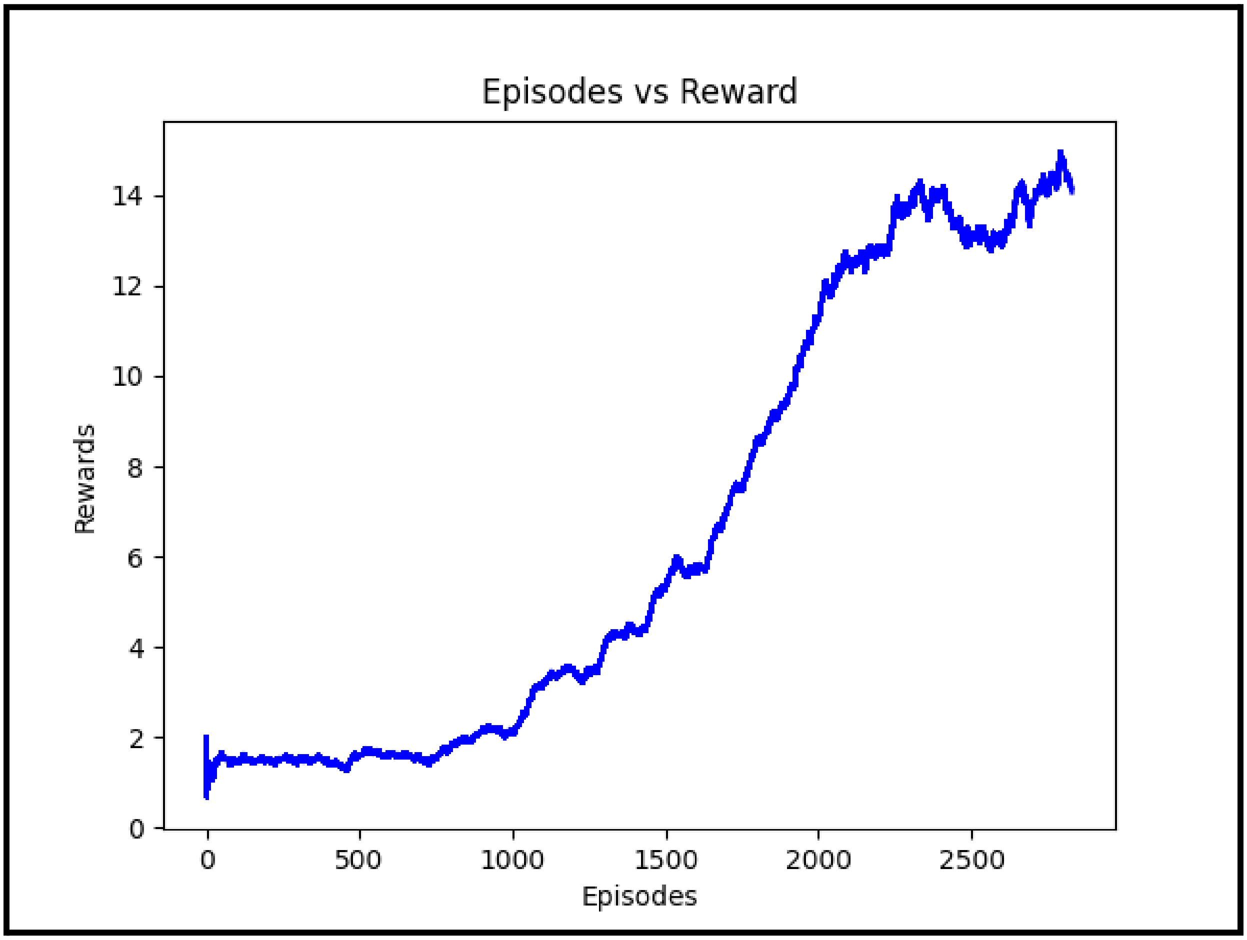
Name(s): Taaha Kazi, Nidhish Kamath
Netid(s): tnkazi2, nkamath5

Mean Reward Reached using DQN and DDQN: DQN: 2.4; DDQN: 14

Uploaded Saved DQN/DDQN Model on Canvas (whichever performs better) : Yes

Uploaded your Agent.py and Agent_double.py file on Canvas : Yes

Plot of Mean Evaluation Reward for the model that reaches the target score (Either DQN or DDQN): Plot of Mean Evaluation Reward for DDQN:



Provide a few sentences to analyze the training process and talk about some implementation details:

1) **We found our implementation to be unstable, i.e. it the same hyperparameters gave different scores over multiple runs; hence we had to restart kernel multiple times to get scores around the expected**
2) **Tried out both of these paradigms, but our score of 14 was achieved in implementation (ii) :**

(i) Where Q-value-next-state-target is chosen based on arg max-ing over the output of the target network.

(ii) Where Q-value-next-state-target is chosen based on the action suggested by the policy network (arg max-ing over the output of the policy network when input is next_state).

3) **Used default hyperparameters (from config)**
4) **Ensured that gradients are not computed for the weights of the policy network when not intended.**
5) **Noticed that PyTorch had two suitable losses we could use, nn.smoothedL1loss and nn.HuberLoss; we used the latter.**