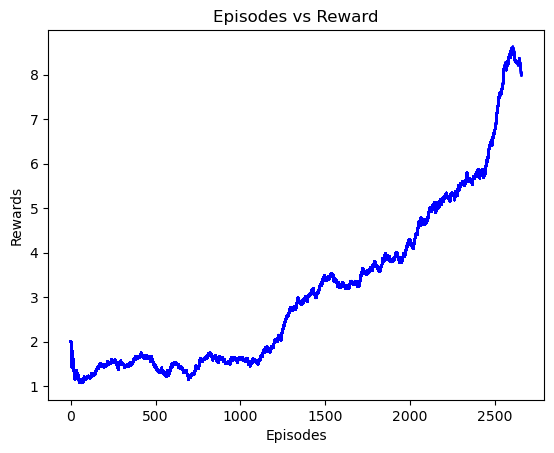
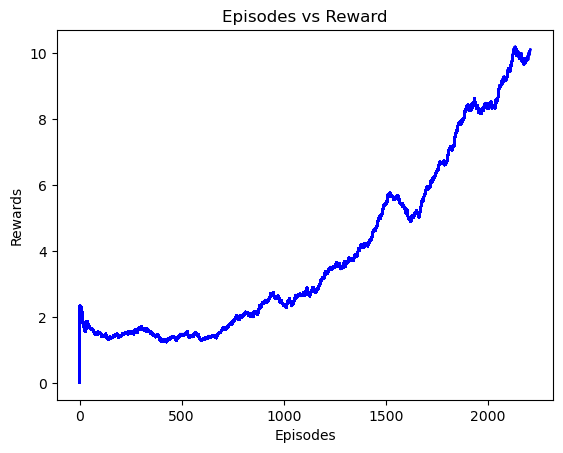
**Name(s):** Vaibhav Raheja

**Netid(s):** vraheja3

**Mean Reward Reached using DQN: 8.04**

**Plot of Mean Evaluation Reward vs. epochs for DQN model: **

**Mean Reward Reached using Double DQN: 10.1**

**Plot of Mean Evaluation Reward vs. epochs for Double DQN model: **

**Uploaded Saved DQN and Double DQN Models on Canvas: Yes**

**Uploaded your Agent.py and Agent\_double.py files on Canvas: Yes**

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

**agent.py and agent\_double.py**

These files contain the main logic for the DQN and Double DQN agents, respectively. Here's how they are structured:

* **Action Selection**: One of the primary functions of these agents is to decide which action to take based on the current state of the game. This decision is made using an ε-greedy policy, where the agent will most often choose the action that it believes has the highest expected reward based on its current knowledge (exploitation), but occasionally it will choose a random action (exploration). This balance helps the agent to discover new strategies.
* **Training Loop**: The agents learn by interacting with the environment. For each action taken, the environment provides a new state and a reward. This information is used to update the agent's knowledge. The update rule depends on the type of agent:
  + DQN: Updates the policy based on a reward prediction error.
  + Double DQN: Uses a second network to reduce overestimations of action values, which can lead to more stable and reliable training outcomes.
* **Memory**: Both types of agents implement a replay memory. This is where past experiences are stored, and then sampled randomly to break the correlation between consecutive learning updates. This helps to improve the stability of the learning algorithm.

**Extra Credit**

**Did you generate the videos for visualizing the agents performance. Yes**

**If you attempted the DQN LSTM Agent, give your implementation details. Yes**

**Mean Reward Reached using DQN\_LSTM: Train Failed**

**Plot of Mean Evaluation Reward vs. epochs for the DQN\_LSTM: Train Failed**

**Provide a few sentences to compare DQN\_LSTM training process with that of DQN and double DQN: Train Failed**

**Answer the questions accordingly if you did the corresponding part. The questions are just prompts. You should elaborate a bit more if you can.**

1. What games did you apply the extra credit to? How does it work?
   1. The game BreakoutDeterministic, using an LSTM-based agent helps it remember and utilize the ball's trajectory and speed over multiple frames. This improves decision-making for paddle positioning, as the LSTM can predict the ball's future movements based on past actions, enhancing gameplay effectiveness.
2. What other algorithm did you use? Explain and cite all your sources. Any issues you got in training your new algorithm.