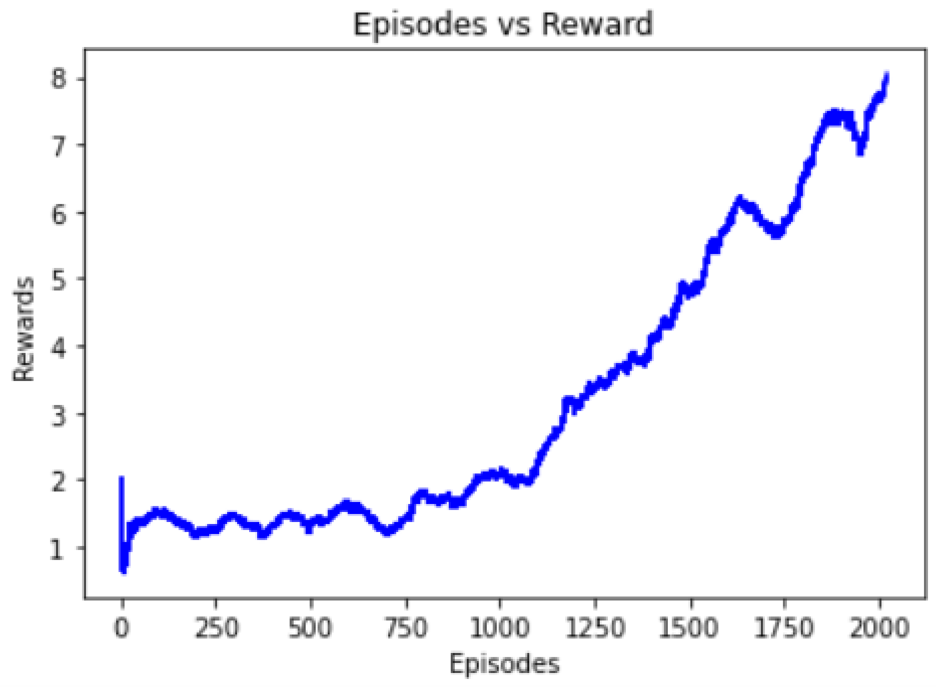
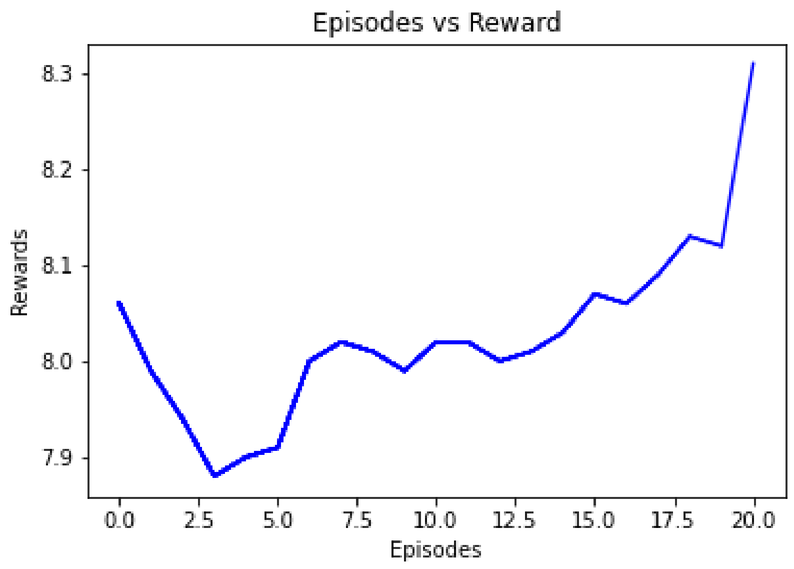
Name(s): Qi Long, Yihong Yang
Netid(s): qilong2, yihongy3

Mean Reward Reached using DQN: 8.31

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



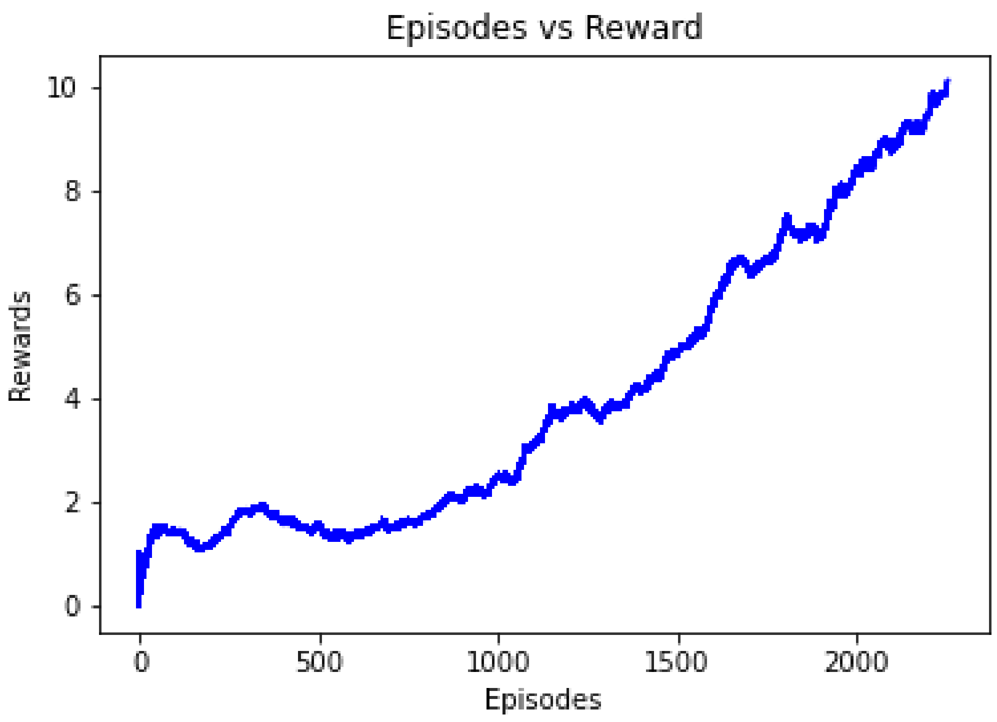
(Reach 8.05 Mean Reward after 2000 epochs)



(Continue training to reach above 8.3 Mean Reward)

Mean Reward Reached using Double DQN: **10.11**

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:

For both DQN and Double DQN, the training process is not continuously increasing the reward, instead, it has some drops in between.

Comparing DQN and Double DQN, Double DQN is more stable than DQN, from plots above, DQN has large fluctuations at around 1750 and 2000 epochs but Double DQN has overall smaller fluctuations.

For implementation, Double DQN takes snapshots every few epochs from DQN network as target.

Extra Credit

Extra Credit 1: DQN-LSTM

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

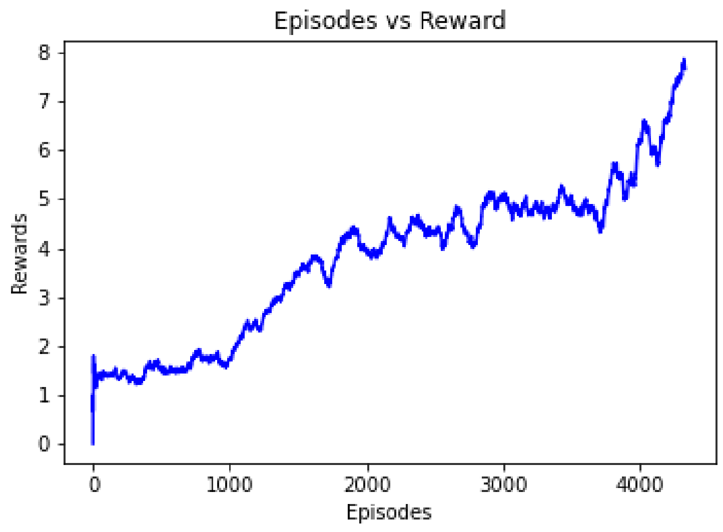
LSTM:
- **Num_layers = 3**

Training:
- **Scheduler_step_size = 300000**
- **Epsilon: slow down the decrement when between 0.1 and 0.3.**
- **Loss Clip: min = -10, max = 10.**

Hyperparameters:
- **Batch_size = 32**
- **lstm_seq_length = 5**

Mean Reward Reached using DQN_LSTM: 8.07

Plot of Mean Evaluation Reward vs. epochs for the DQN_LSTM:



Provide a few sentences to compare DQN_LSTM training process with that of DQN and double DQN:

The training process is more sensitive to epsilon greedy sampling weight. As shown from training plot, DQN_LSTM makes fastest progress after around 3500 epochs, when epsilon is close to minimal value 0.01. The training process before that is flat compared to DQN and Double DQN.

Besides, one layer of LSTM is not good enough, instead, 3 layers of LSTM is better. This results in higher training cost, so we slow down the learning rate and epsilon decrement.

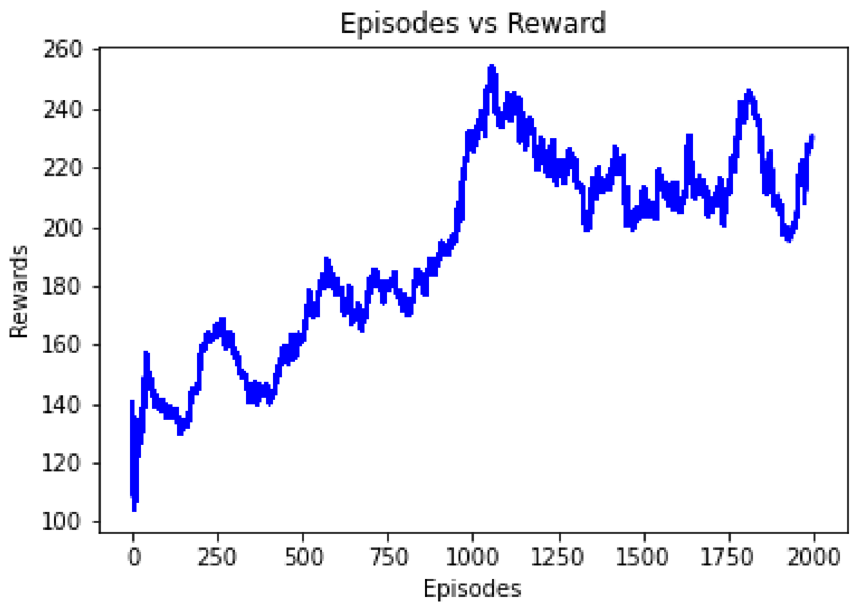
Extra Credit 2: Demon Attack

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?
2. What other algorithm did you use? Explain and cite all your sources. Any issues you got in training your new algorithm.

Atari Demon Attack
Action space: move left, move right, fire.
Reward: hitting enemy aircraft.
Lives: 3
Implementation: DQN
Hyperparameters:

- **Batch_size = 32**



Since demon stack has gradual increment in difficulty, after getting reward of about 250, the scenario will change, which is not reached often by model so not good performance after reward above 250.

Overall, the performance of DQN on Demon Attack is good. It can reach rewards above 250. Since the reward range is larger than Breakout, it takes more time to train the model and the training process is more unstable.

Extra Credit 3: Video Generation

Implementation: code in MP5.ipynb and MP5_DQNLSTM_EC.ipynb

1. **Gym Render:**
   - **Package: RecordVideo**
   - **Render_mode: 'rgb_array'**
2. **Agent:**
   - **Epsilon = 0**
   - **HISTORY_SIZE = 1 for LSTM, others 4**
   - **Process same as training process with small modifications.**

Experiment: videos are attached in submission

- **DQN**
- **Double DQN**
- **DQN-LSTM**
- **DQN for Demon Attack**