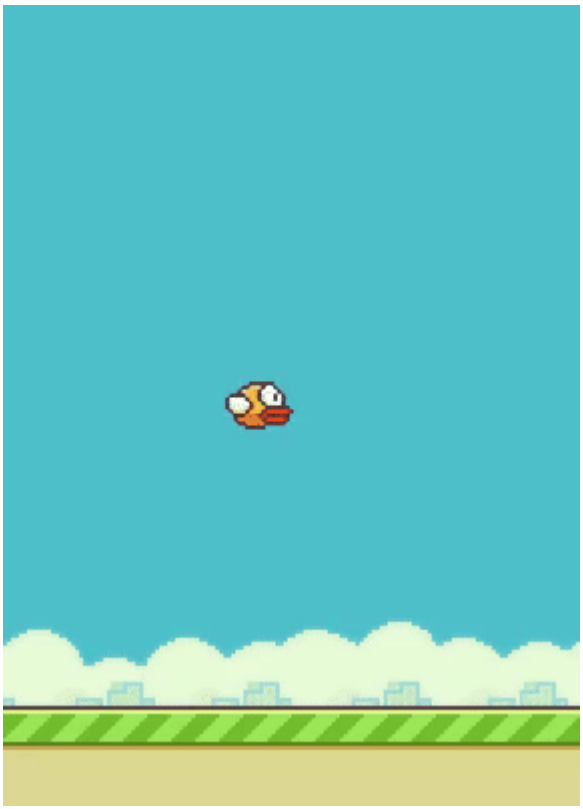
ME 599 - HW 1 Notes

For this HW I will be exploring a third party environment (Flappy Bird) along with Deep Q Learning (DQN).



Environment

Installation

1. Install Gymnasium via the notes on their Github:

   ```
   pip install gymnasium
   ```

2. Install the third party flappy-bird-environment:

   ```
   pip install flappy-bird-environment
   ```

3. Test the environment by playing flappy bird:

   ```
   python -m flappy_bird_env
   ```

4. Get started with

```
import flappy_bird_env
env = gymnasium.make("FlappyBird-v0")
```

Environment state space, action space, and rewards

The action space is binary and very simple. Either the bird flaps or doesn't. A value of 0 means no action and a value of 1 means the bird will flap.

The state space is simply the RGB image that players see while playing the game. It is given via a numpy array of shape (800, 576, 3) where entries are between 0 and 255.

Rewards are given as follows:

- +0.001 for each frame the bird is alive
- +1 for each pipe the bird passes through

Learning to fly: DQN

Note: My notebook largely follows and adapts PyTorch's DQN tutorial found here.

Reasoning for choosing DQN:

- The state space is large and given as an image
- A simple Q table would be massive and infeasible
- DQN has a hope of interpreting the image and learning a policy

Downsides of DQN:

- Expensive: Needs many evaluations of NN to learn a policy
- Not interpretable: No clear way to understand how the NN is learning and making decisions

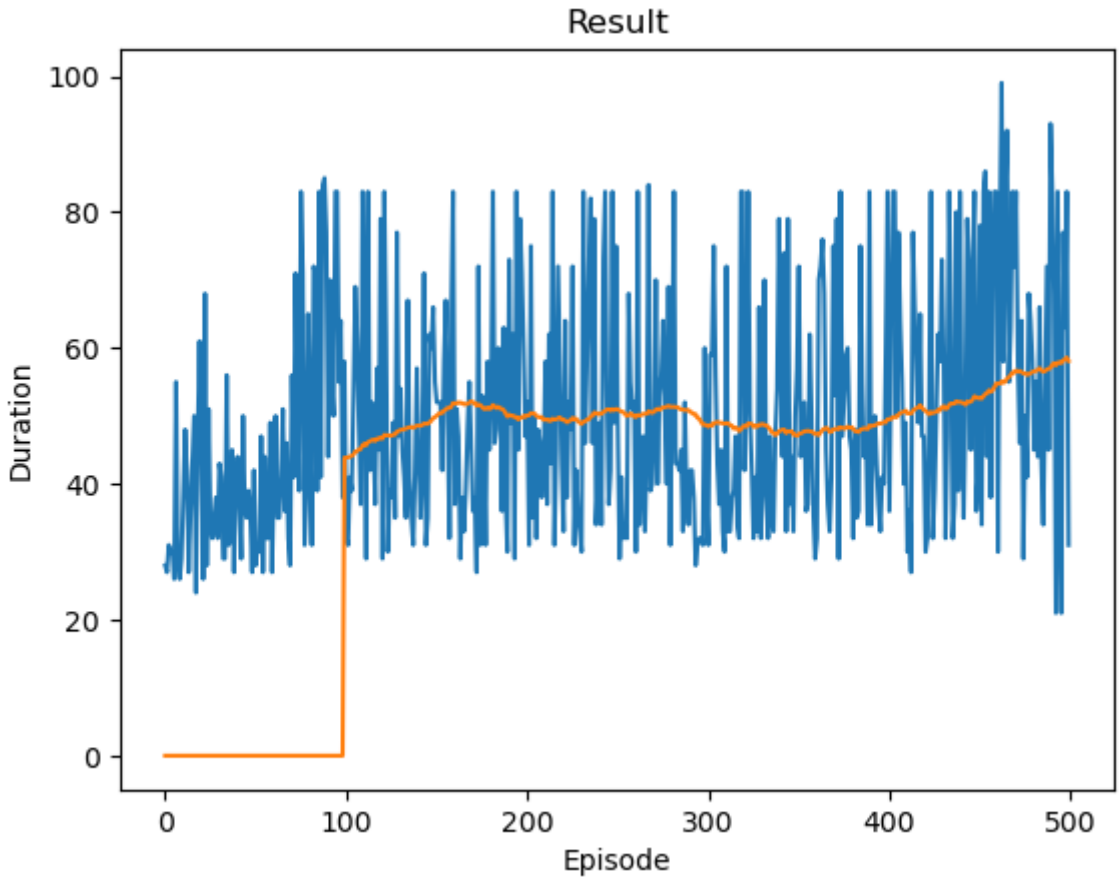
Key components of implementation:

- Create and train a NN to approximate the Q function
- Experience replay: Store past experiences and sample from them to train the NN. The downside is that it requires more memory which my gpu can't handle 😦
- Preprocessing the state space is very important. I first convert it to grayscale, then downsample the image by a factor of 4. This greatly reduces the required memory needed for training.
- NN size was expanded to have 5 layers with 100 nodes each. Still a fully connected network

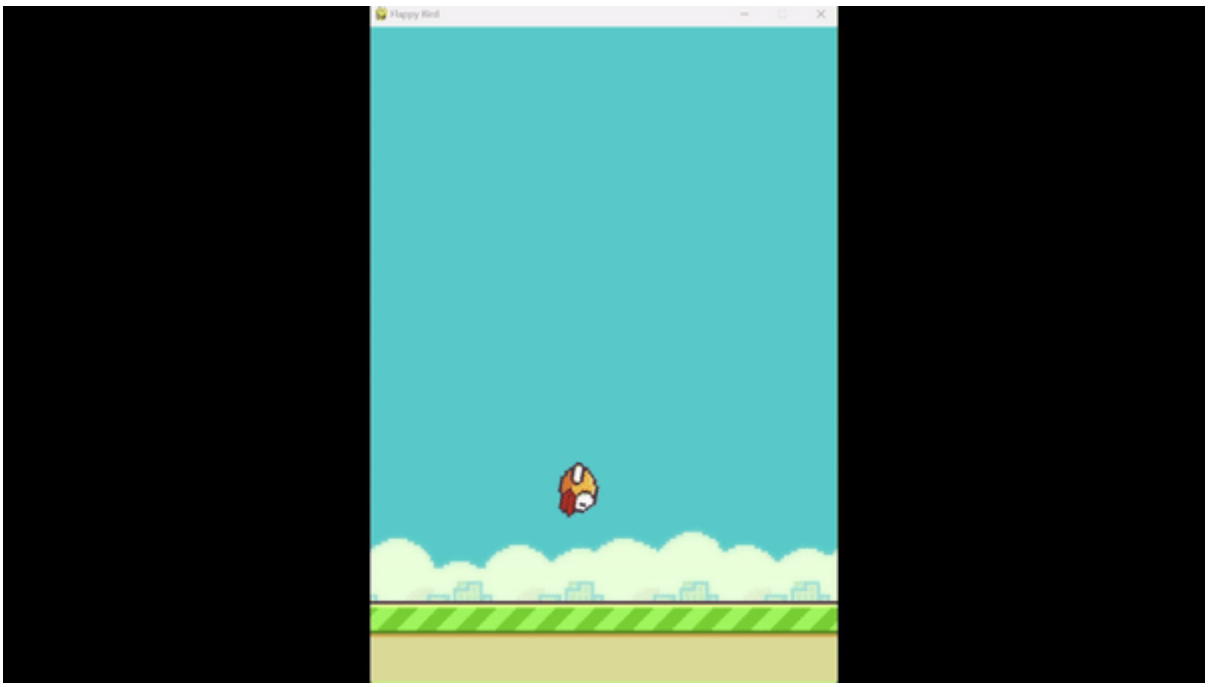
Results

I found that my implementation had a memory leak of sorts and often crashed after some amount of episodes. However, using pre-processing significantly reduced the memory requirements and allowed me to train for longer.

Below is a plot of the duration that the bird survived for each training episode. Although noisey, there is a clear upward trend in the bird's survival time and hence the DQN's ability to act in the environment. However, the majority of the training consisted of the agent simply learning to keep the bird in bounds and pipes tended to be a hard stop for the agent.



Below is a give of several episodes of the agent playing the game. The results are clearly noisey and the agent often fails entirely to move, but there are some instances where the agent is able to stay in bound and approaches the pipes.



Far more learning (I only trained for ~8 min) and a larger network would likely be needed to help the agent learn to navigate the pipes. Furthermore, a network tailored to images (such as a convolutional neural network) would likely be more effective in this environment.