Flappy Bird Project Discussion

Chris Minkwon Choi

Implementation

The Flappy Bird project is divided into two parts, the env.py and dqn.py. The env.py is where the environment including bird, pipes, and floors are created, and the dqn.py is where the training and learning takes place.

env.py

The implementation of the environment comes in two fold: reset and step. The reset would create the initial environment and step will take care of processing each step, starting from the resetted frame.

- Reset: First, the variables are initialized, such as velocity, list of pipes, bird positions, and observation list. Then using for-loop, the 4 frames and 50x50 matrix is filled with '0's. Then the floor and ceiling is drawn with '1's. Then the helper function "drawBird" is called where the function would draw a bird of '2's in given y position. The drawBird first deletes any bird that exists now and draws a new bird.
- Step: step function also starts with initializing the variables. Then each frame is pulled forward such that frame 2 becomes frame 1. Then frame 3, which is the frame that this function will be mainly working on, goes through a light reset. Instead of creating a new frame, a loop iterates through all 50x50 spaces and fills it with '0's. Now the order of drawing is important. Later on, the collision detection is done as the bird is drawn, thus all other objects in the game including the pipe have to be drawn before the bird is drawn. First, the floor and ceiling is drawn. At this time, the implementation of static environment, which does not move, is completed.

  The pipe is implemented with a list. Inside the list, each pipe is stored in the form of a list with two elements. The first element is the height of the pipe and the second element is the position of the pipe. For example, pipes = [[4,15],[9,45]] shows two pipes, each with height of 4, at position 15 and height of 9 at position 45. After each pipe is moved a pixel forward, a loop goes through the list of pipes to check if it needs to delete or spawn new pipes. If the newest pipe's location is below a given pixel, it will spawn a new pipe. If the oldest pipe's location is below -5 (gives enough time for thickness 7 pipe to disappear), then the pipe is removed. After updating all pipes they are drawn on the board.

  Now the bird is drawn. But each time a bird is about to change its current number to '2', it checks what is the current number. If the current number is 1, this means the bird has collided with the pipes or boundaries of the map. Thus the collision is detected, changing reward to -1 and done to True. After the bird is drawn, then the position of pipes are checked to give reward. If the location of a pipe is 23, it means the bird has passed that pipe without collision. Thus the reward is set to 1.0. Then the speed of the bird is changed according to the given action. At last, the observation is stored in the current state variable and returns observation, reward, and done.

dqn.py

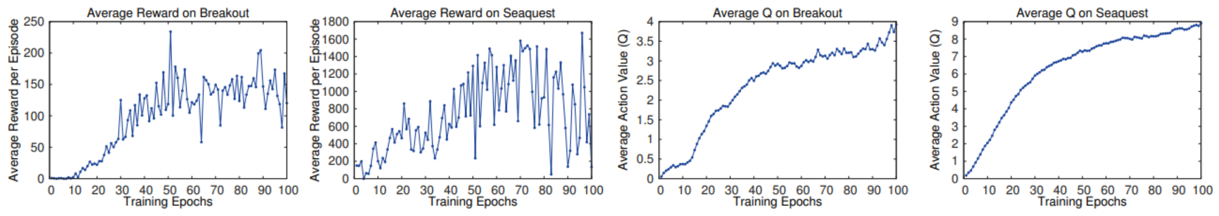
The DQN implantation was relatively simple. First a random number between 0.01 and 1.00 is created. If the number is smaller than epsilon, then a random action is chosen. If the number is greater than the epsilon, then the action corresponding to the maximum q_value is used. Then the step is run and the result is returned to next_state, reward and done variables.

Plot and Analysis

At this part, I will plot two graphs using episodes as the x axis, q-value and average steps as y axis. Then I will compare these graphs to the ones from DQN paper.

DQN

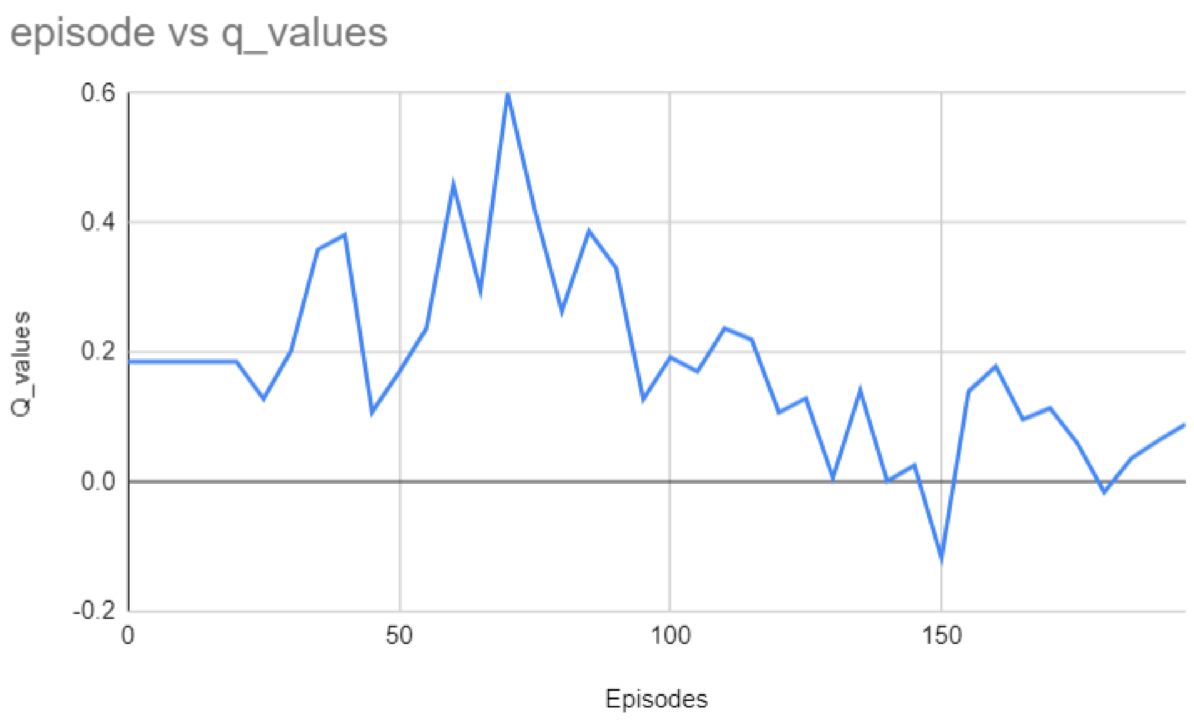
The following is graphs from the DQN paper.



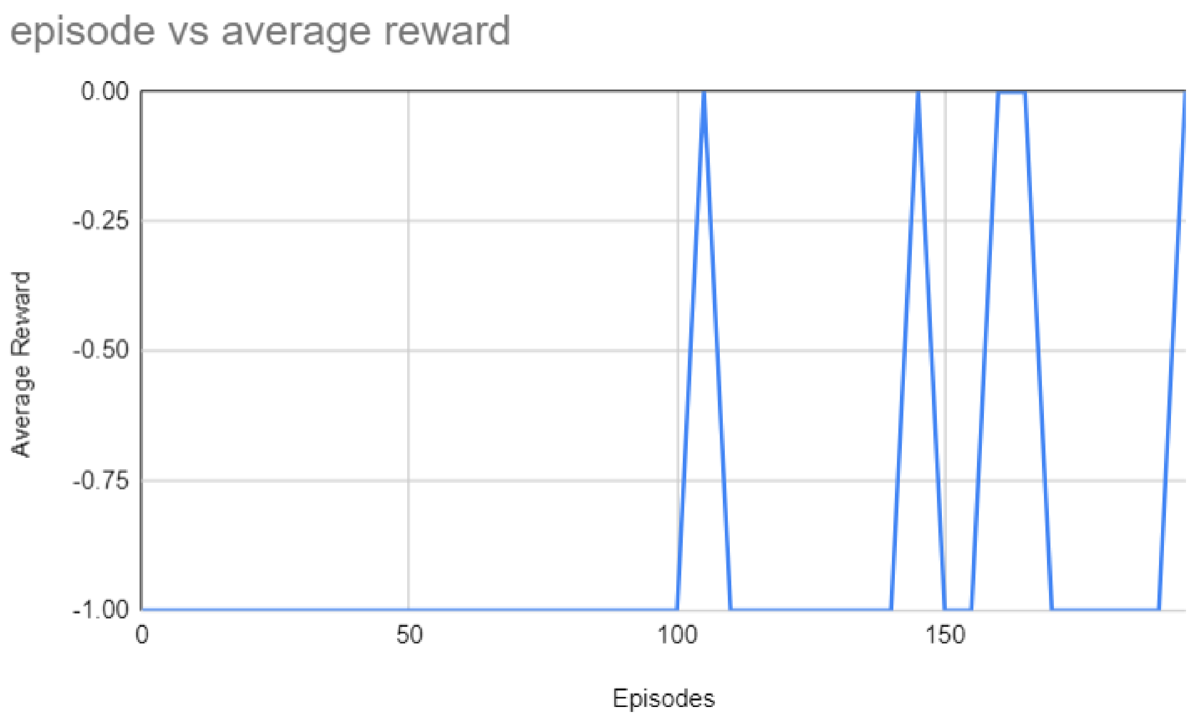
Looking at the two average reward charts, the graph has high noise. The graph on Seaquest has such a bad noise that the trend at later training epochs is nearly undeterminable. However, the one on Breakout shows a better increasing trend as training epoch increases. Despite all noises, we can see a general trend of increasing average reward as the episode continues. On the right two graphs, the graph of average Q value and episodes shows steady increase with slight noise on both games. One critical difference between the graphs from DQN paper and my flappy bird project is the training time. We only trained for 7500 batches when DQN paper trained for 5 million batches. Considering this difference, I focused on the general trend instead of the exact same results.

Flight

The following is a graph of the flight data, where the bird is trained to fly without pipes.

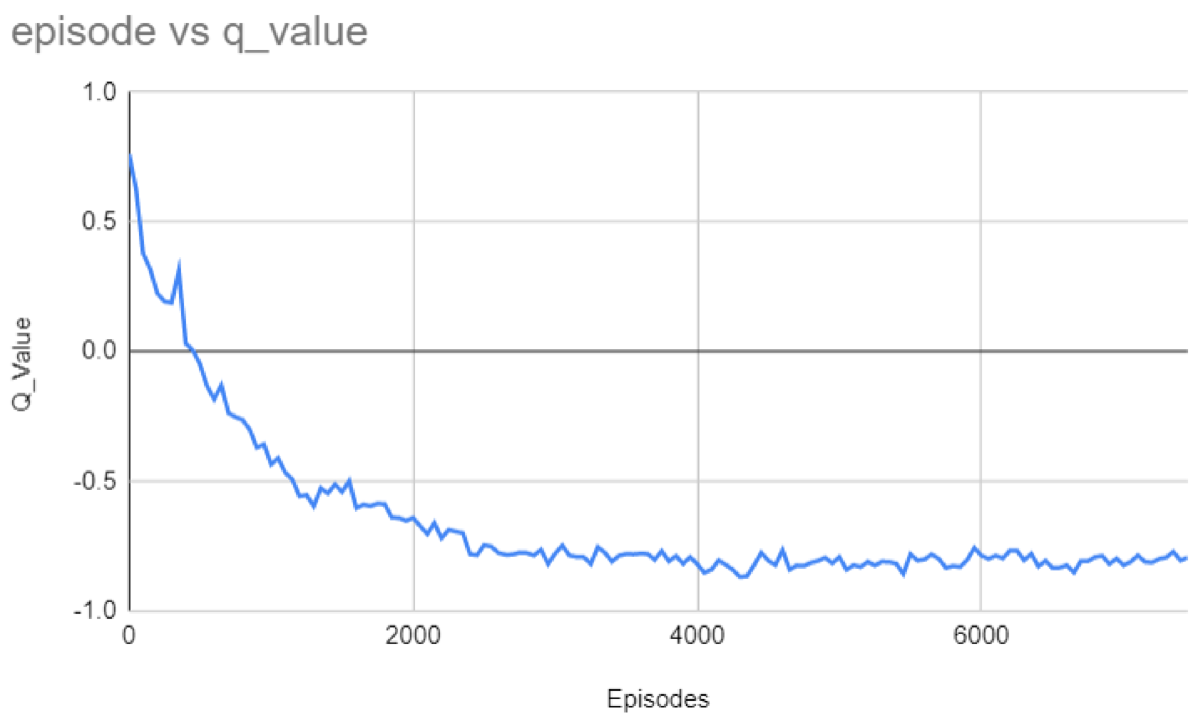


As there are no pipes, there is no way for the bird to get reward. Thus the 0.0 is the best possible result that this bird can get. The best graph would show a straight line along the x-axis. In the graph, the q_value moves closer to the x-axis as expected as the episode increases. Thus this graph shows the AI is successfully learning how to fly.



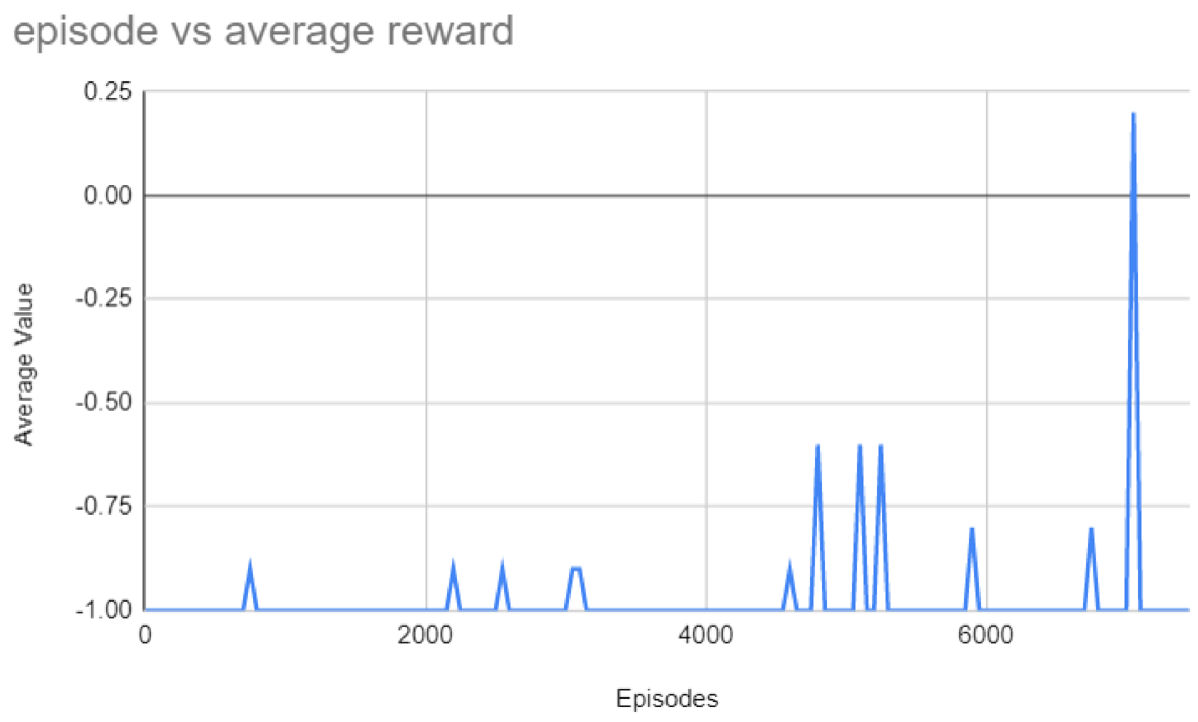
Looking at the graph, most of the episodes ended with an average reward of -1. This is because since there are no pipes, there is no way for the bird to gain rewards. On some occasions, the bird won 0 average reward because it made it to the testing limit, 250, without a crash. The occurrence of 0 reward increases as the training continues. For the first 100 episodes, there was no case that the bird reaches 250, but as it moves closer to later episodes, it reaches 250 steps more frequently. At 160 and 165, the bird reached 250 steps in a row. Thus this graph can be a proof that the bird learned to fly.

Pipes
Now the bird is tested with pipes. The graph showing the relationship between the result and q_value follows.



Similar to the result shown in the flight graph, there is a rapid decreasing trend at the beginning, where the AI is trying very random ideas to start. Then the AI settles and starts to learn. From episode 4000, there is a slight increasing trend in Q_value. Looking at the Q_value graph of Breakout in the DQN paper, there is a section at low episodes where the Q_value has very slow increase. Considering that DQN paper used 5 million tries and I had 7000 tries, this can be the starting portion of what is also shown in DQN paper.

The graph showing the relationship between the result and average reward follows.



This graph is showing the increasing trend. At the beginning, the average value was at -1.0, and as training continues, there are more and more higher rewards, and finishes off with a surprising 0.2. The graph does not look exactly same as the DQN paper, but the trend is similar.

Extra Discussion

The AI did show a similar trend to the one of DQN paper, it did not reach the limiting step or close to 100 steps. As the result of the training was not satisfactory, I ran multiple tests and training to check where it was wrong. After that, I manipulated the environment to see if this would make any difference.

Suspected Lists

The following is a list of suspected areas and how I tested them. All of the following areas are proven to be not the cause of the problem.

- Reward system
    - Printed current reward every step, and observed. The reward system was not the problem. The program accurately printed out '0' when there is no collision or going through a pipe, '-1.0' when crashed, and '1.0' when the bird successfully passed a pipe.
    - Also tried to change the way 1.0 is given. The new method was to check the position of the pipe, instead of the bird. When the position of the pipe moves behind the bird without game ending, this means the bird successfully passed the pipe, thus rewarded a 1.0. This method also worked well, but did not change the problem.

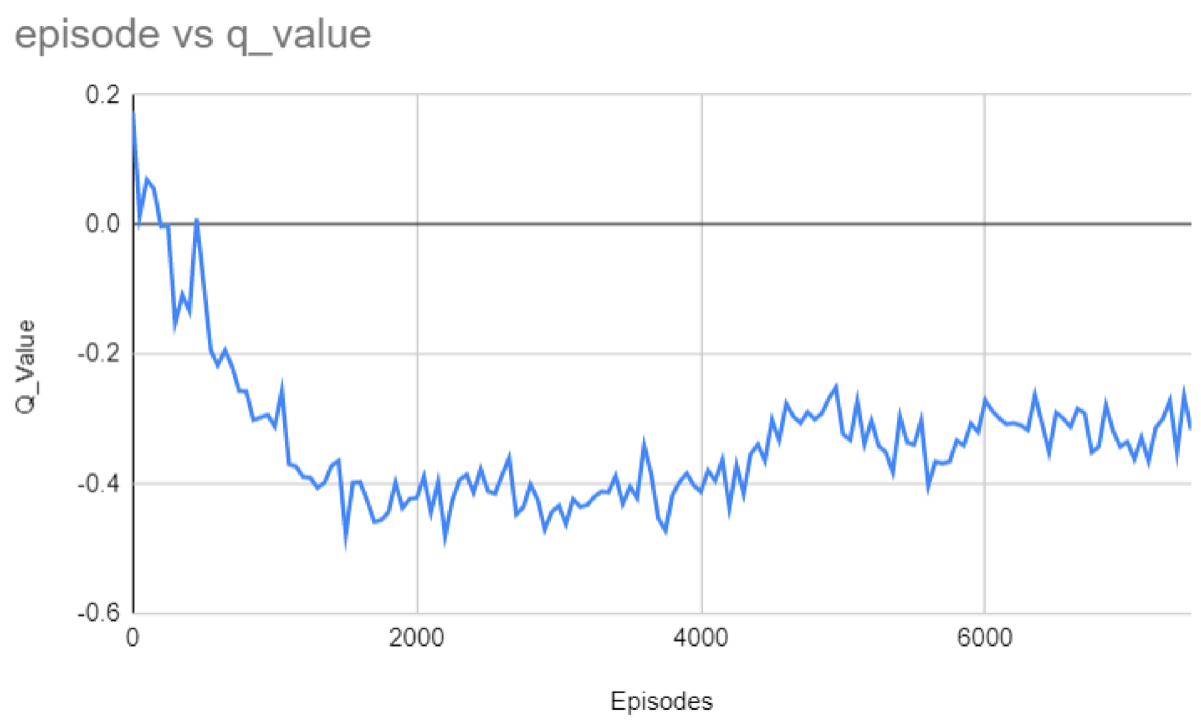
- Storing past frames
    - This was brought up by Jack.
    - I used copy.deepcopy to copy the numbers, not the reference.
    - Tried to completely fill all 4 frames with 0s and refill, but did not make a difference
    - Used the debugging tools of PyCharm to inspect each frame, and confirmed that the past frames are well recorded.
- Pipe generation and degeneration
    - Count the number of pixels between each pipe and the gap between top and bottom pipes. Confirmed that the environment is well generated as required.
    - Confirmed the pipes that have already passed are deleted
- Is the environment playable
    - Ran the demo with pipes, and it showed a playable environment as required by the instruction.
    - Confirmed that the bird had enough time to go through the pipe, but crashed.
- Simply not learning at all
    - This was suspected when I had -1.0 for most flight results, but after running the test again with minor fixes, I received 0 and 1, and extremely rarely 3s. Thus the bird actually is learning.

After confirming all suspected areas are correctly functioning, I designed an experiment. I believe the training was not enough or the environment is too harsh. To test this hypothesis, enlarged the gap between top and bottom pipe, creating an easier environment. If the result is similar to the one from DQN paper, then this proves that the environment was set up correctly, but simply given conditions were too difficult.

Experiment with larger gap

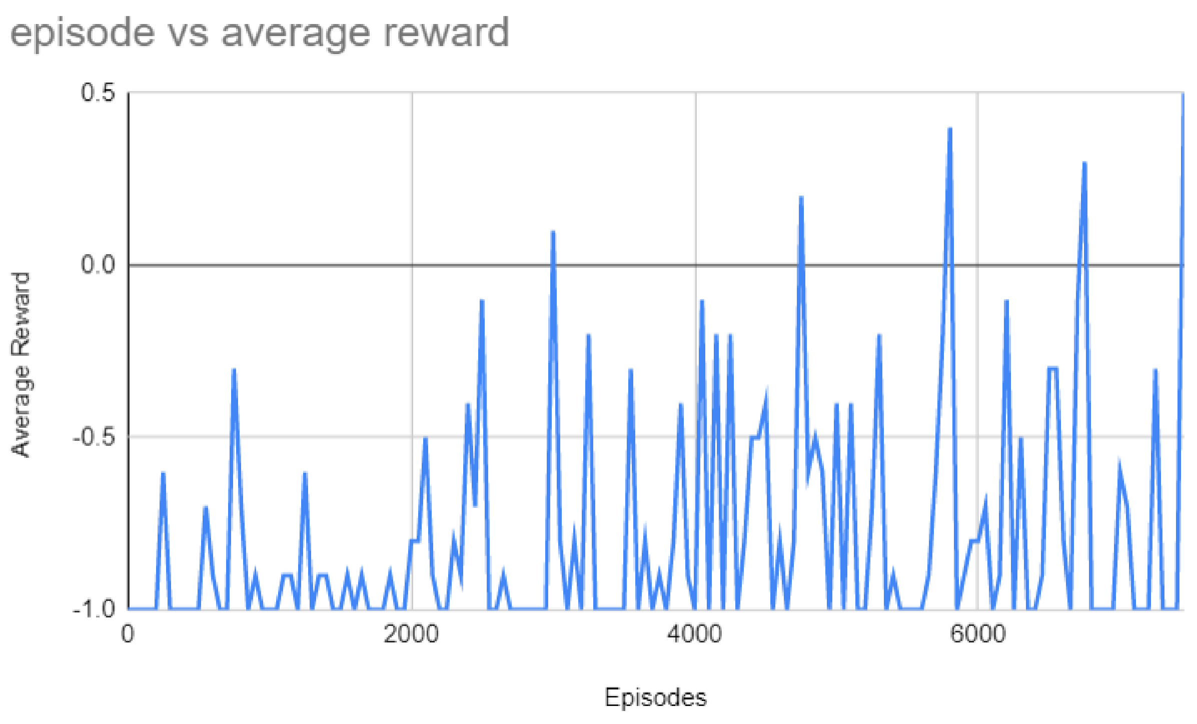
This time, the gap between top and bottom pipe is increased by 10 pixels, such that the total gap is 30 pixels. With this change, I ran the same training, and the following is the graph from this experiment results.

First is the graph showing the relationship between the episodes and q_value.



The graph shows rapid decrease at the beginning to about 1500 episodes. This is an expected phenomenon where the bird is starting from almost completely random choices and moving on to a steady learning stage. From the 1500th episode, there is a steady increase in the q value. The q_value is expected to continue to increase if we had more episodes. Compared to the DQN paper, the graphs are showing a similar positive slope, which shows the training is working.

The following is a graph showing average reward.



Just like the DQN paper, there is lots of noise, but there is some information achieved through this experiment. There are two positive results. First, there is a steady increasing trend. As the training continues, the average reward is increasing, as well as the highest average reward record. Second, the frequency of -1.0 reward is decreasing. -1.0 average reward shows that through that whole 50 episode, none of the birds passed the first pipe. Thus having less than -1.0 average record is a very promising result.

As this experiment shows better accuracy and closer match with the DQN paper, I concluded that the environment was on the correct track. I spent three entire weeks solving this problem, but I could not solve it. Instead, I discussed where I might have failed and what I have tried to solve this question.

Thank you Thomas for a great PA, and thank you all TAs for helping me in countless OHs.

Attaching the log from this experiment.

	avg_reward	best_reward	episode	q_vals	avg_steps	best_steps
0	-1	-1	0	0.1749604791	7	7
1	-1	-1	50	0.01386384573	10.9	22
2	-1	-1	100	0.06926655769	6	6
3	-1	-1	150	0.0550596863	6.5	7
4	-1	-1	200	-0.002901136642	7.9	9

5	-0.6	1	250	-0.00183964392 6	20.2	28
6	-1	-1	300	-0.1512289941	13.9	20
7	-1	-1	350	-0.1084903553	9.2	13
8	-1	-1	400	-0.1331411153	13.2	19
9	-1	-1	450	0.009335836396	16.1	21
10	-1	-1	500	-0.0902659744	16.6	24
11	-0.7	1	550	-0.1940490454	12.5	36
12	-0.9	0	600	-0.2166806459	19.1	29
13	-1	-1	650	-0.1941319406	15.2	22
14	-1	-1	700	-0.2191938013	15.2	21
15	-0.3	2	750	-0.2567066848	31.4	52
16	-0.7	1	800	-0.257392168	26.6	53
17	-1	-1	850	-0.3013804555	15.3	21
18	-0.9	0	900	-0.2975590527	10.3	29
19	-1	-1	950	-0.2934762537	6.8	10
20	-1	-1	1000	-0.3110805452	11.7	21
21	-1	-1	1050	-0.2561627328	17.2	22
22	-0.9	0	1100	-0.3693737388	10.7	29
23	-0.9	0	1150	-0.3730036914	18.1	30
24	-1	-1	1200	-0.3890028298	11.9	18
25	-0.6	1	1250	-0.3904363811	13	28
26	-1	-1	1300	-0.40590325	10.4	16
27	-0.9	0	1350	-0.3981451392	15.7	40
28	-0.9	0	1400	-0.3719385564	8.3	29
29	-1	-1	1450	-0.3644353151	8.6	10
30	-1	-1	1500	-0.478782177	6	6
31	-0.9	0	1550	-0.3982579112	12.6	47
32	-1	-1	1600	-0.3970701694	9.7	16
33	-0.9	0	1650	-0.4252867997	19	36
34	-1	-1	1700	-0.4584857523	16.4	22
35	-1	-1	1750	-0.4557953179	7.1	15
36	-1	-1	1800	-0.444070071	14.9	22
37	-0.9	0	1850	-0.3982851207	11.8	30
38	-1	-1	1900	-0.436955303	12.6	21
39	-1	-1	1950	-0.423073858	14.6	16
40	-0.8	0	2000	-0.4213597178	18.2	33
41	-0.8	0	2050	-0.3888262212	17.2	44
42	-0.5	1	2100	-0.4441015422	24	57

43	-0.9	0	2150	-0.3971055746	10.4	35
44	-1	-1	2200	-0.4799838662	6.5	10
45	-1	-1	2250	-0.425042212	6	6
46	-0.8	0	2300	-0.3938958645	16.4	45
47	-0.9	0	2350	-0.3855931163	17.6	33
48	-0.4	1	2400	-0.4131436646	19.8	43
49	-0.7	1	2450	-0.37676543	17.8	58
50	-0.1	2	2500	-0.410894841	32.5	52
51	-1	-1	2550	-0.4151226878	6	6
52	-1	-1	2600	-0.3864135146	11	11
53	-0.9	0	2650	-0.3596995175	17.8	29
54	-1	-1	2700	-0.4471573234	7.9	21
55	-1	-1	2750	-0.4360096753	6	6
56	-1	-1	2800	-0.4008207619	9.5	11
57	-1	-1	2850	-0.4241419435	8.2	11
58	-1	-1	2900	-0.4699672461	12.7	21
59	-1	-1	2950	-0.4433331788	13.2	22
60	0.1	2	3000	-0.4335501194	44.7	95
61	-0.8	1	3050	-0.4619640112	25.3	68
62	-1	-1	3100	-0.4234641194	9.2	15
63	-0.8	0	3150	-0.4354773462	14.9	34
64	-1	-1	3200	-0.4324010909	6	6
65	-0.2	1	3250	-0.4196656048	30.1	65
66	-1	-1	3300	-0.411780864	11.2	22
67	-1	-1	3350	-0.4128777683	7.6	16
68	-1	-1	3400	-0.3873612583	7.5	21
69	-1	-1	3450	-0.4307340682	8.7	21
70	-1	-1	3500	-0.4042547047	6	6
71	-0.3	1	3550	-0.4216974974	21.4	44
72	-1	-1	3600	-0.3403399289	16.9	21
73	-0.8	0	3650	-0.3852597773	14.5	42
74	-1	-1	3700	-0.4529354572	6	6
75	-0.9	0	3750	-0.4718993902	13.3	43
76	-1	-1	3800	-0.4163586199	11.2	19
77	-0.8	0	3850	-0.3965711892	14.2	40
78	-0.4	1	3900	-0.3835875094	27	37
79	-0.9	0	3950	-0.4025653303	21.2	43
80	-1	-1	4000	-0.4121323228	7	10
81	-0.1	3	4050	-0.3791640699	36.2	113

82	-1	-1	4100	-0.394890368	6.5	7
83	-0.2	2	4150	-0.3629252911	32.9	85
84	-1	-1	4200	-0.4360491633	6.1	7
85	-0.2	1	4250	-0.3702631295	31.5	53
86	-1	-1	4300	-0.4149184227	6	6
87	-0.8	1	4350	-0.353962332	23.2	52
88	-0.5	1	4400	-0.3393181264	23.6	54
89	-0.5	1	4450	-0.3644600213	27.2	55
90	-0.4	1	4500	-0.3012246192	33.3	68
91	-1	-1	4550	-0.3328705728	12.5	27
92	-0.8	0	4600	-0.2759819329	14.8	37
93	-1	-1	4650	-0.2956312895	6.9	15
94	-0.8	0	4700	-0.3060204983	12.9	37
95	0.2	1	4750	-0.2890796959	32.4	46
96	-0.6	1	4800	-0.300475955	19.4	46
97	-0.5	0	4850	-0.2913980484	25.7	47
98	-0.6	0	4900	-0.2681871057	20.2	46
99	-1	-1	4950	-0.2513742447	6	6
100	-0.4	1	5000	-0.3231047988	28.3	52
101	-1	-1	5050	-0.3321123123	7.9	15
102	-0.4	1	5100	-0.2715111375	30.1	53
103	-1	-1	5150	-0.3382711411	6	6
104	-1	-1	5200	-0.3022800684	8	16
105	-0.7	0	5250	-0.3414302468	18.4	42
106	-0.2	1	5300	-0.3506741524	34.6	55
107	-1	-1	5350	-0.3813087642	6	6
108	-0.9	0	5400	-0.2958200276	18.5	33
109	-1	-1	5450	-0.3358929455	7	7
110	-1	-1	5500	-0.3396731317	14.7	25
111	-1	-1	5550	-0.3019068837	6	6
112	-1	-1	5600	-0.4025816917	6.7	7
113	-0.9	0	5650	-0.3656630516	16.9	44
114	-0.6	1	5700	-0.3685054183	27.6	82
115	-0.2	1	5750	-0.3654867113	31.8	77
116	0.4	1	5800	-0.3330731392	45	65
117	-1	-1	5850	-0.3407844305	9.3	10
118	-0.9	0	5900	-0.3068472445	13.3	30
119	-0.8	1	5950	-0.3197211325	21.3	28
120	-0.8	0	6000	-0.2711754739	11.9	33

121	-0.7	0	6050	-0.2873610556	19.5	31
122	-1	-1	6100	-0.299076736	7	12
123	-0.9	0	6150	-0.3078728318	10.5	30
124	-0.1	1	6200	-0.306178242	37.3	54
125	-1	-1	6250	-0.3095754683	7.6	13
126	-0.5	2	6300	-0.3166811764	31.6	87
127	-1	-1	6350	-0.2629753947	10.1	15
128	-1	-1	6400	-0.3065700829	9	12
129	-0.9	0	6450	-0.3509866893	14.1	31
130	-0.3	0	6500	-0.2899338305	35.8	45
131	-0.3	1	6550	-0.2990078032	32.8	53
132	-0.8	0	6600	-0.3118858635	19.3	49
133	-1	-1	6650	-0.2841482759	11.9	21
134	-0.1	3	6700	-0.2909708023	36	126
135	0.3	3	6750	-0.3513604701	48.9	127
136	-1	-1	6800	-0.3426294327	8.4	10
137	-1	-1	6850	-0.2784822583	14.2	22
138	-1	-1	6900	-0.3178687096	13.1	24
139	-1	-1	6950	-0.3426391184	9.1	13
140	-0.6	0	7000	-0.3352178037	27.2	49
141	-0.7	0	7050	-0.3619036376	15.5	33
142	-1	-1	7100	-0.3284255266	6.6	7
143	-1	-1	7150	-0.3649305105	9	15
144	-1	-1	7200	-0.3135798872	18.9	23
145	-0.3	1	7250	-0.299197495	29.7	53
146	-1	-1	7300	-0.2716866136	6.9	7
147	-1	-1	7350	-0.3517868817	9.4	15
148	-1	-1	7400	-0.2649862468	8	12
149	0.5	1	7450	-0.3164232671	37	61