Pong Game AI

– Kai Chen(kc3041), Yixiong Wu(yw2914), Zijun Nie(zn2146), and Suer Hu(sh3589)

1. Project in brief

In this project, we try to train a ping pong player who learns the direction and distance of optimal movement through:

- A six-layered Convolutional Neural Network to recognize ball behavior from updated screen input
- A reinforcement learning system which targets the learning agent at and against behaviors e.g. Catching the ball, avoiding loops

Game rules:

- The ball is served by the opponent AI first, at randomd angle; Upon wining, the serving right is passed to the opponent
- Score in testing (shown on pygame screen) is calculated as one additional points if opponent fails to catch the ball

File structure - Project.zip contains:

- A *pygame_player.py* for overall functions
- A *pong.py* for defining opponent AI and screen display configurations
- A *pong_player.py* for defining example class for playing pong which imports from *pygame_player.py*
- A *main_train.ipynb* for training the learning agent
- A *main_test.ipynb* for testing and demonstrating agent performance

2. Our design

Opponent player - We designed the opponent pong player to be competent while vulnerable:

- Competent: In order for it to catch most balls, we make the opponent cheat by
  - reading the ball's direction from coordinates
  - aligning the ball's speed with its own speed
- Vulnerable:
  - The opponent's action is defined at a random rate
  - Due to limited speed, it cannot catch certain servings e.g. when the ball heads from upper half towards upper corner while it is in the lower half of the field

RL process:

- Observation VS exploring - We settled on 100k observations steps and 1000k exploring steps. To balance the exploration/exploitation tradeoff, we raised the observation steps after finding our agent trapped in a local optimal policy to only move downwards
- Learning reward - We designed the reward system to enhance several actions:

- For the first catch in a round, whichever player catches the ball is rewarded 3 points. We encourage our agent to catch more serve with this design
- For continuing catch in the round, reward is given at $3*0.6^{catch times}$. We discourage our agent to be trapped in tacit agreement to stay put and make every catch

CNN network:

- The structure is based on DeepMind
- We improved it with RELU, adding one maxpooling layer for stability, and adding one dropout layer against overfitting

3. Improvement

(Baseline)

Features:

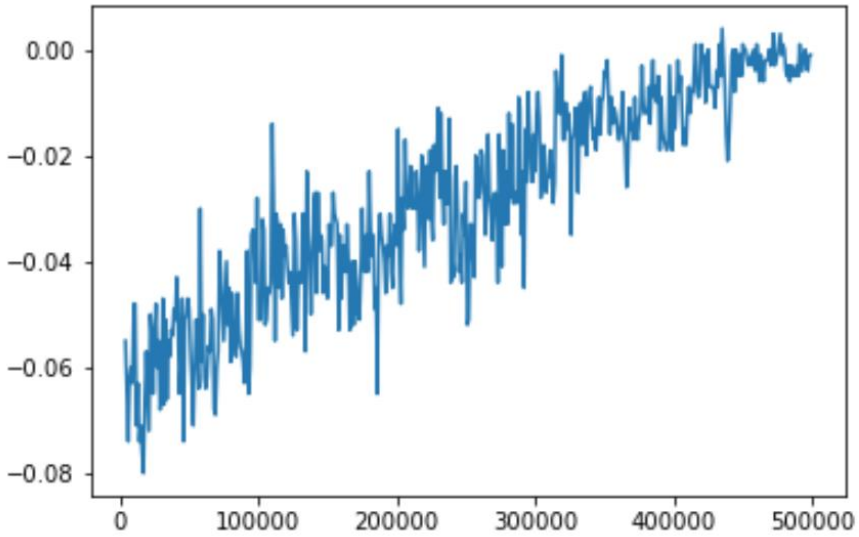
- Basic reward (1 point if opponent fails to catch)
- 2-directioned serve (ball starts from the middle line 45 degrees to one of the four corners)
- Basic Network(6-layered)
- 20k observation steps

Performance (at 500k steps):

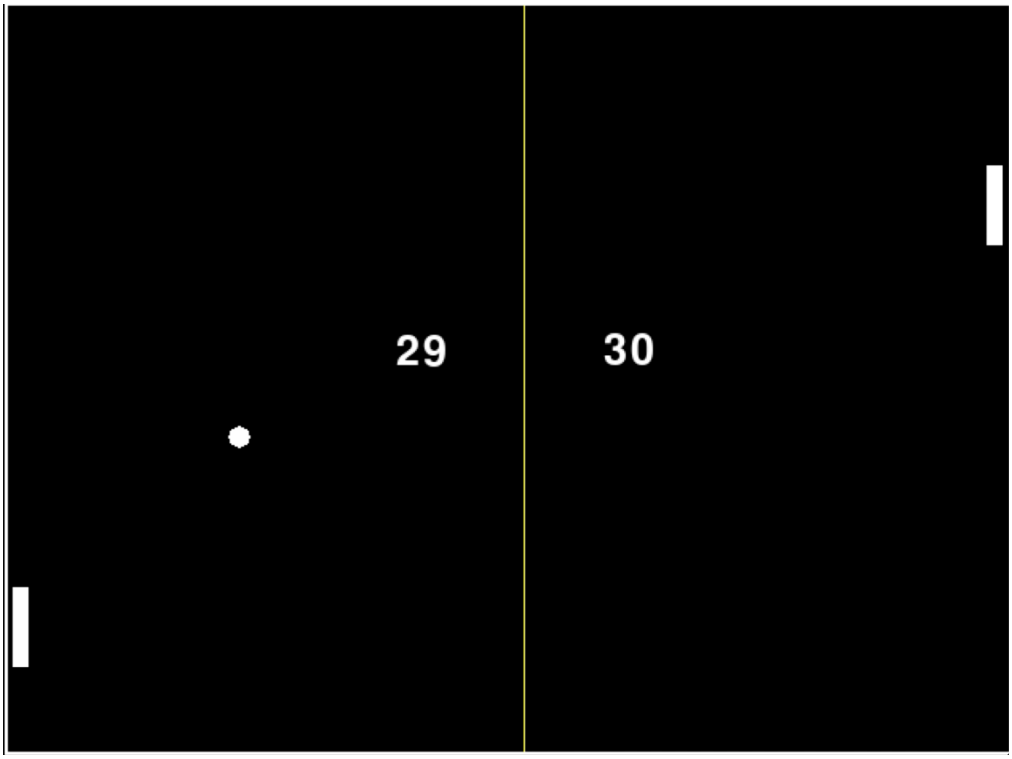
- According to the reward we printed at each step (i.e. either side gets one point whenever the opponent fails to catch a serve), this baseline model learns the best strategy after 50,0000 training steps.

```
Training - Time: 3000; Reward: -0.055
Training - Time: 4000; Reward: -0.064
Training - Time: 5000; Reward: -0.074
Training - Time: 6000; Reward: -0.062
Training - Time: 7000; Reward: -0.06
Training - Time: 8000; Reward: -0.063
Training - Time: 9000; Reward: -0.048
Training - Time: 10000; Reward: -0.048
Training - Time: 11000; Reward: -0.071
Training - Time: 12000; Reward: -0.063
Training - Time: 13000; Reward: -0.074
Training - Time: 14000; Reward: -0.071
Training - Time: 15000; Reward: -0.076
Training - Time: 16000; Reward: -0.08
Training - Time: 17000; Reward: -0.066
Training - Time: 18000; Reward: -0.057
Training - Time: 19000; Reward: -0.064
```

```
Training - Time: 484000; Reward: -0.003
Training - Time: 485000; Reward: -0.005
Training - Time: 486000; Reward: -0.004
Training - Time: 487000; Reward: -0.005
Training - Time: 488000; Reward: -0.003
Training - Time: 489000; Reward: -0.005
Training - Time: 490000; Reward: 0.001
Training - Time: 491000; Reward: -0.004
Training - Time: 492000; Reward: -0.004
Training - Time: 493000; Reward: -0.001
Training - Time: 494000; Reward: 0.0
Training - Time: 495000; Reward: -0.003
Training - Time: 496000; Reward: -0.004
Training - Time: 497000; Reward: -0.001
Training - Time: 498000; Reward: -0.001
Training - Time: 499000; Reward: -0.004
Training - Time: 500000; Reward: -0.001
```



- Video performance is recorded at https://drive.google.com/open?id=1OvXVqnOP3i5ZNfkGtW7FN4bTQcV-PSYY



Problems:

- This model performs well to catch balls coming from one direction, but fails to come up with a good strategy for balls from the other direction.
- Too simple rules: Despite satisfactory overall training process, the model is too simplified as the ball only comes from two possible directions. Thus, the model will underperform in other cases.

Code: Code is accessible at https://github.com/KaiChenColumbia/MagicBalls/tree/master/model_base

Model 1

Improvement:

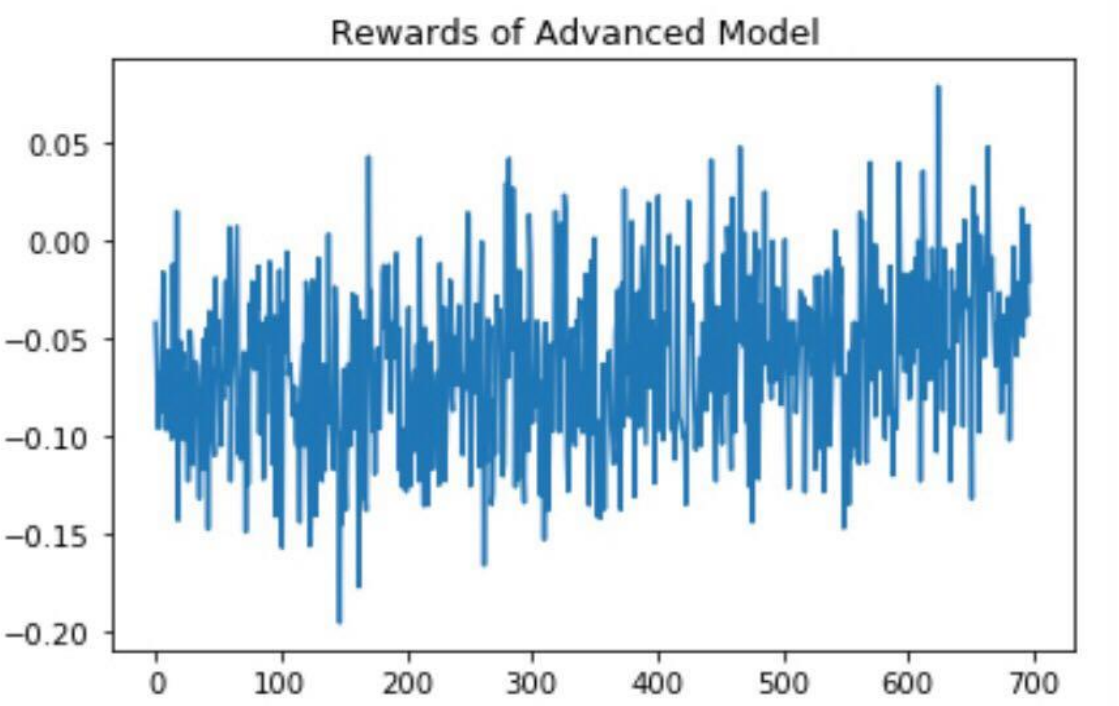
- To raise the difficulty of the game and allowing more complex cases, we changed the serving to random directions, in addition to features of the baseline model.

Performance (at 700k steps):

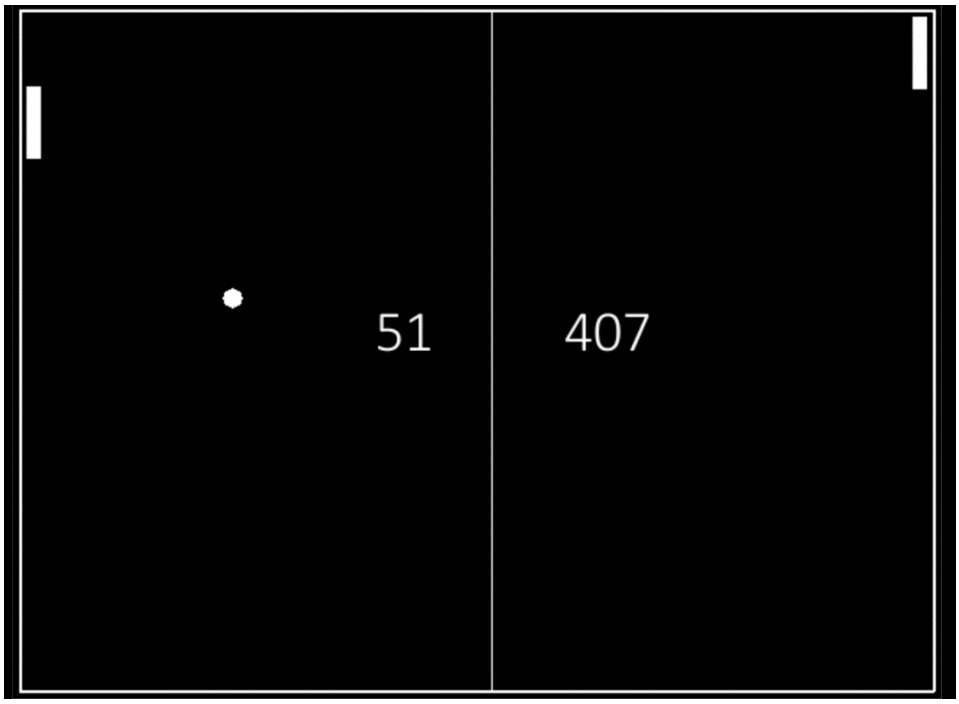
- According to reward we printed at each step, which is one point whenever the opponent fails to catch a serve, our agent is learning how to gain the most scores at a positive but slow speed.

```
Time: 200; Reward: -0.0608333333333
Time: 300; Reward: -0.03
Time: 400; Reward: -0.06
Time: 500; Reward: -0.04
Time: 600; Reward: -0.03
Time: 700; Reward: -0.09
Time: 800; Reward: -0.04
Time: 900; Reward: -0.08
Time: 1000; Reward: -0.07
```

```
Time: 14000; Reward: -0.05
Time: 14100; Reward: 0.0
Time: 14200; Reward: -0.06
Time: 14300; Reward: -0.06
Time: 14400; Reward: -0.01
Time: 14500; Reward: -0.04
Time: 14600; Reward: -0.03
Time: 14700; Reward: -0.09
Time: 14800; Reward: -0.06
Time: 14900; Reward: 0.01
```



- Video performance is recorded at https://drive.google.com/open?id=1exXiiL2s7iMSdHrc8GToFHk9dTFoaUx
-



Problems:

- Local optimal: We noticed that during the training, the bar was easily trapped in a local policy of only moving downwards/upwards, which requires a large amount of cases of moving up and gaining reward to cancel out.
- Not amazing performance: Although the game is set simple at our first try, the learning curve is not significant.

Code: Code is accessible at https://github.com/KaiChenColumbia/MagicBalls/tree/master/model

Model 2

Improvement: In terms of the above problems, we -

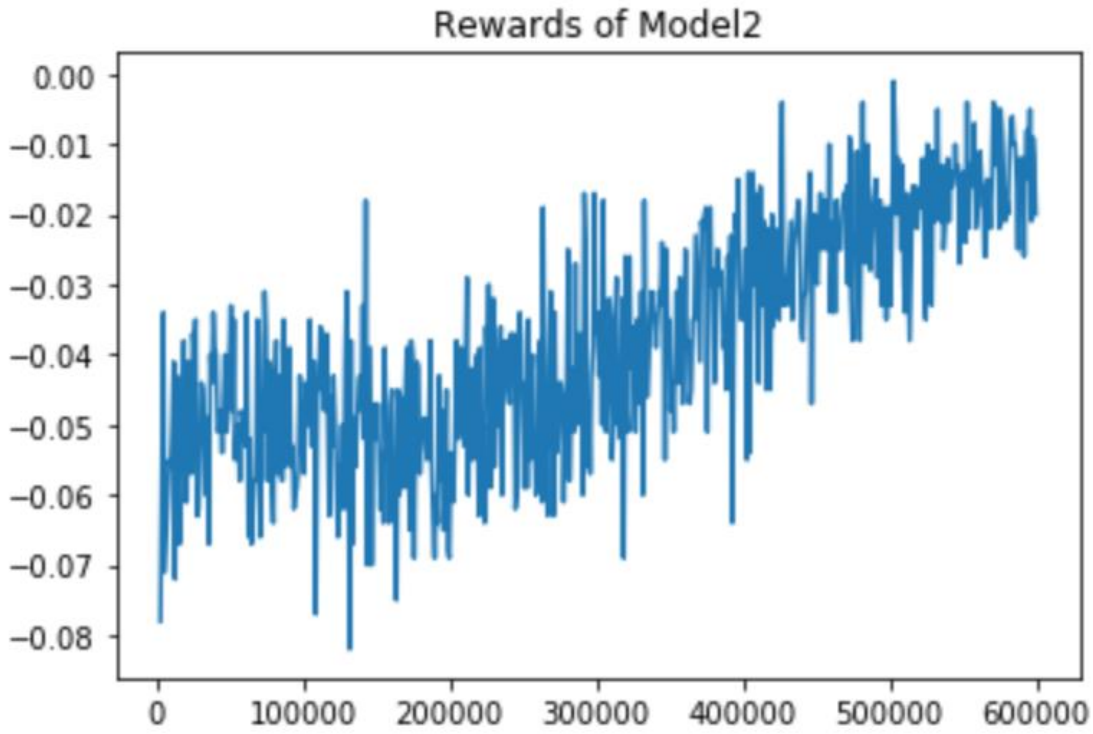
- increased the number of observation steps to 50000, and created checkpoints every 50000 steps to check how much performance has improved
- raised the difficulty of the game by serving the ball at random angle
- improved CNN to adopt DeepMind structure

Performance (at 600k steps):

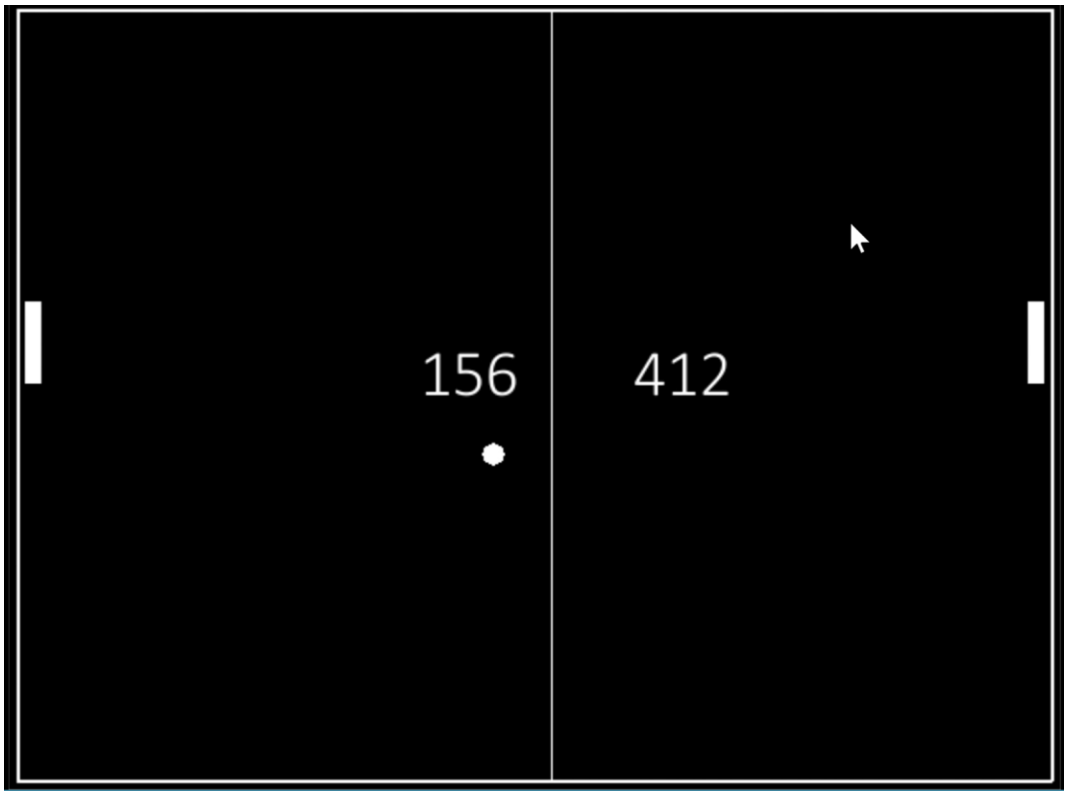
- Since we have a more powerful CNN structure to recognize ball movement, the learning curve is significantly steeper than the baseline model, despite the more difficult training process.

```
Observation - 98000 / 100000
Observation - 99000 / 100000
Observation - 100000 / 100000
Training - Time: 2000; Reward: -5.064
Training - Time: 3000; Reward: -0.078
Training - Time: 4000; Reward: -0.056
Training - Time: 5000; Reward: -0.034
Training - Time: 6000; Reward: -0.071
Training - Time: 7000; Reward: -0.066
Training - Time: 8000; Reward: -0.062
Training - Time: 9000; Reward: -0.055
Training - Time: 10000; Reward: -0.056
```

```
Training - Time: 590000; Reward: -0.018
Training - Time: 591000; Reward: -0.012
Training - Time: 592000; Reward: -0.026
Training - Time: 593000; Reward: -0.013
Training - Time: 594000; Reward: -0.008
Training - Time: 595000; Reward: -0.015
Training - Time: 596000; Reward: -0.005
Training - Time: 597000; Reward: -0.021
Training - Time: 598000; Reward: -0.009
Training - Time: 599000; Reward: -0.012
Training - Time: 600000; Reward: -0.02
```



- Video performance is recorded at https://drive.google.com/open?id=1urqt67ETAMmA52moPIF5oTbHycFk0qib



Problems: New problems emerged –

- The two bars reached a “consensus” to stay in diagonal corners with which minimal movement ensures no one loses the game, this resulted in no update of the policy. So we need to penalize the length of every round.
- The agent still misses a lot of ball at the first serve, so we need to put reward on the actual hit of the ball, in addition to the opponent’s failure.
- Though learning rate is higher, the best performance still shows a reward below 0. We considered it to be the result of harder rules/stronger opponent thus fewer positive reward.

Code: Accessible at https://github.com/KaiChenColumbia/MagicBalls/tree/master/model2

Model 3 (Final version)

Improvement: In terms of the above problems, we –

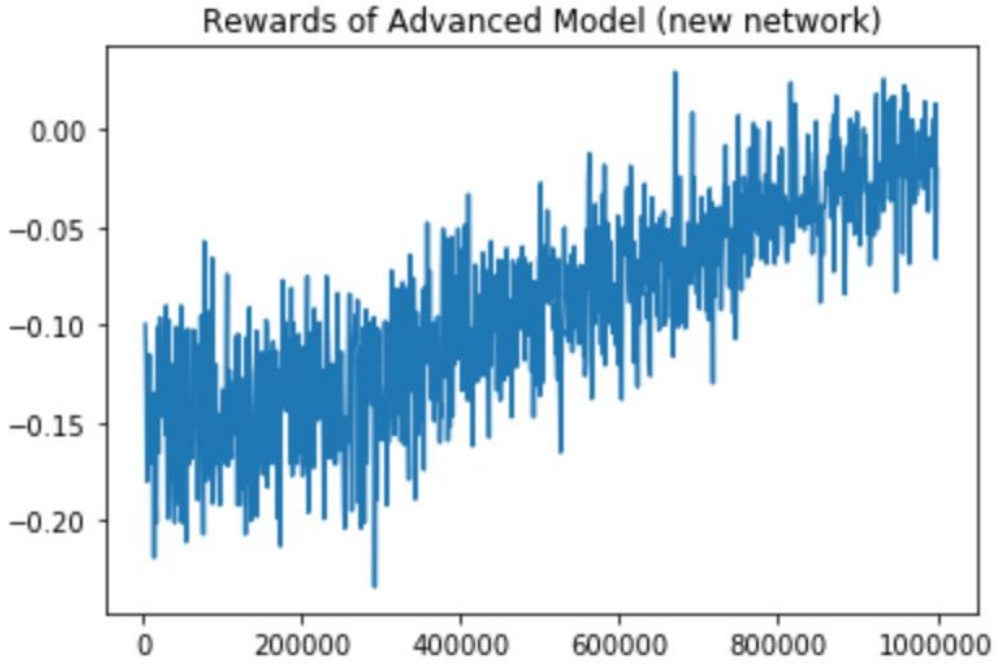
- Updated reward system to give 3 points to whoever hits the ball
- updated reward system to penalize repeated trajectory of the ball movement. Namely, reward will experience an exponential decay with a rate of 0.6 every time it hits the bar in a specific round, and the loop is forced to break after 10 hits
- upgraded CNN with RELU

Performance (at 1000k steps):

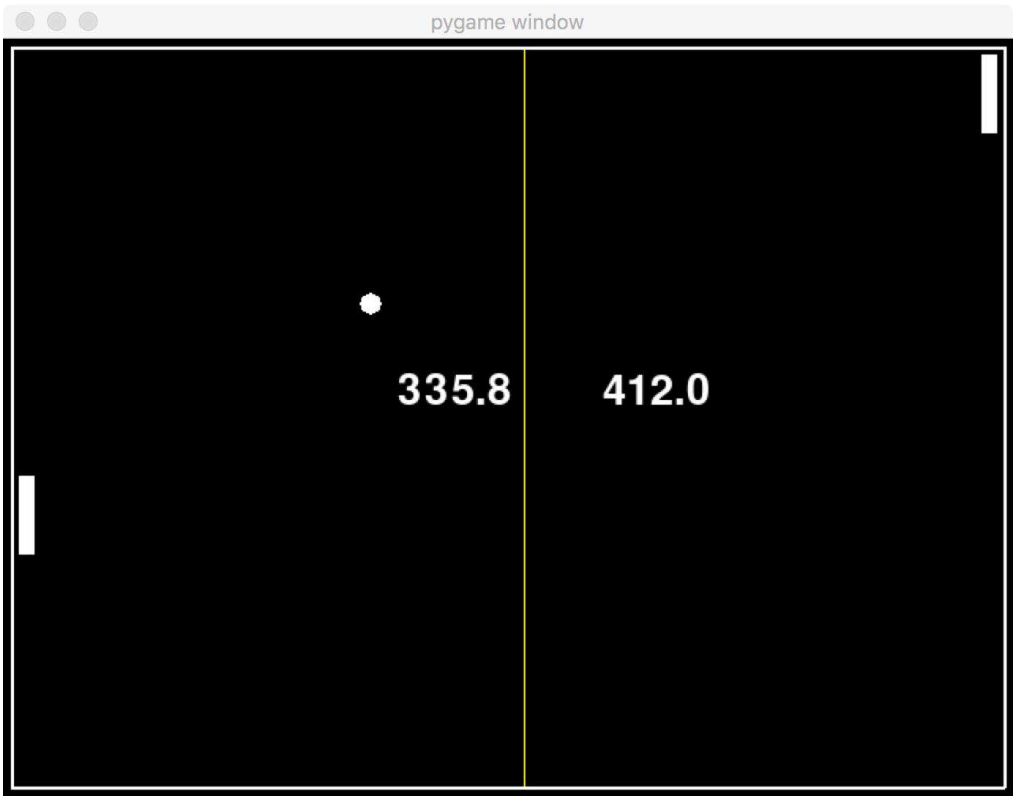
- Learning curve is steep after we upgraded the CNN, and the agent can almost reach an average reward of 0 by the end of our 1000k steps training, which means it can almost be as good as its cheating opponent. Video record shows that our agent is near to getting the same score as its opponent.

```
Observation - 95000 / 100000
Observation - 96000 / 100000
Observation - 97000 / 100000
Observation - 98000 / 100000
Observation - 99000 / 100000
Observation - 100000 / 100000
Training - Time: 2000; Reward: -15.5701
Training - Time: 3000; Reward: -0.0996
Training - Time: 4000; Reward: -0.1226
Training - Time: 5000; Reward: -0.18
Training - Time: 6000; Reward: -0.138
```

```
Training - Time: 990000; Reward: -0.004
Training - Time: 991000; Reward: -0.015
Training - Time: 992000; Reward: -0.0054
Training - Time: 993000; Reward: -0.0184
Training - Time: 994000; Reward: -0.0074
Training - Time: 995000; Reward: -0.0186
Training - Time: 996000; Reward: 0.0056
Training - Time: 997000; Reward: -0.041
Training - Time: 998000; Reward: 0.0134
Training - Time: 999000; Reward: -0.066
Training - Time: 1000000; Reward: -0.0202
```



- Video performance is recorded at https://drive.google.com/open?id=1othJyEePWy1O4lo9Et4UJgV19X6oJd4Z



Code: Accessible at https://github.com/KaiChenColumbia/MagicBalls