Hockey Game

Group 9

B05502087 王竑睿　B06902041 吳采耘　B06902104 吳由由

Work Distribution

- B05502087王竑睿
  - Proposal
  - Pygame
  - Report
- B06902041吳采耘
  - Pygame
  - Poster
  - Code review
- B06902104吳由由
  - Proposal
  - Architecture design
  - Training model

Motivation and Background

- Hockey Game is a kind of well-known competitions in the world. Human players use different strength to form various speed such as smash. We are interested in whether AI could learn to do this.
- Q-learning is a classical method of reinforce learning. However, the state space of Hockey Game is too big for Q-table to handle. Therefore, we use Deep-Q as an alternative.

Problem

- Let AI agent play with rule based agent
- Capture Lots of game screenshot and feed into Convolution Neural Network and collect the training data
- Decreasing exploring rate
- Trying different reward with respect to particular action

Solution

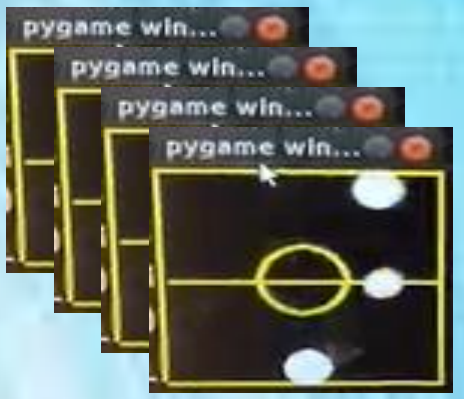
- Let AI agent play with rule based agent
  - Rule based agent always move toward the ball, which means it goes left when the ball is at left and goes right when the ball is right
  - The speed of rule base agent is as fast as the ball. Therefore, it can always catch up the ball. Only the smash ball could get a point.

```
def updateEnemy(bx,by,x,y):
    if x+player_initial_radium<=bx+ball_initial_radium:
        x+=player_initial_speed
    elif x+player_initial_radium>=bx+ball_initial_radium:
        x-=player_initial_speed
    if x<player_initial_radium:
        x=player_initial_radium
    if x>screen_width-player_initial_radium:
        x=screen_width-player_initial_radium
    return [x,y]
```

Solution

- Capture Lots of game screenshot and feed into Convolution Neural Network and collect the training data
  - We keep collecting the tuple (input_screenshot, best_action, reward, new_screenshot) as training data and contain it in a queue.
  - The first 50000 iterations we only collect the data. After the iterations, we started to sample a data from the queue and use it to update the weight of CNN.
  - Formally, the CNN is minimizing the loss function：
    - $\mathrm{L}_i(\theta_i) = \left(y_i - Q(s, a; \theta_i)\right)^2$
    - $\mathrm{y}_i = r + \gamma * \max_{a'} Q(s', a'; \theta_{i-1})$
    - θ_i : the weights of CNN in the i′th iteration r: reward γ: $learning\ rate$
    - $y_i\ is\ our\ target\ and\ Q(s, a; \theta_i)\ is\ the\ output\ of\ CNN$

Solution



- Our CNN model
  - Scale the input screenshot to 84x84, and input 4 to CNN each time.(84,84,4)
  - Using filter1 [8,8,4,32] with stride 4 and get a small pictures (21,21,32)
  - Do 2x2 max pooling and get (11,11,32) images
  - Using filter2 [4,4,32,64] with stride 2 and get a smaller pictures (6,6,64)
  - Do 2x2 max pooling and get (3,3,64) images
  - Using filter3 [3,3,64,64] with stride 1 and get a smaller pictures (3,3,64)
  - Do 2x2 max pooling and get (2,2,64) images
  - Reshape it to 256x1 vector
  - Do fully connected Relu and get another 256x1 vector
  - Matrix multiplication to get 5x1 vector (means:[up, down, left, right, stay])

Solution

- Decreasing exploring rate
  - We let the agent explore the environment with possibility as ϵ. That is, sampling with uniform distribution and behave with policy when the result is higher than ϵ, otherwise behave randomly to explore the environment.
  - We decrease the exploring rate ϵ with linear annealing to make the model act more accurately as the iteration and the data becomes more and more enough.

$$\epsilon = \epsilon - (e_s - e_f)/\text{explore}$$

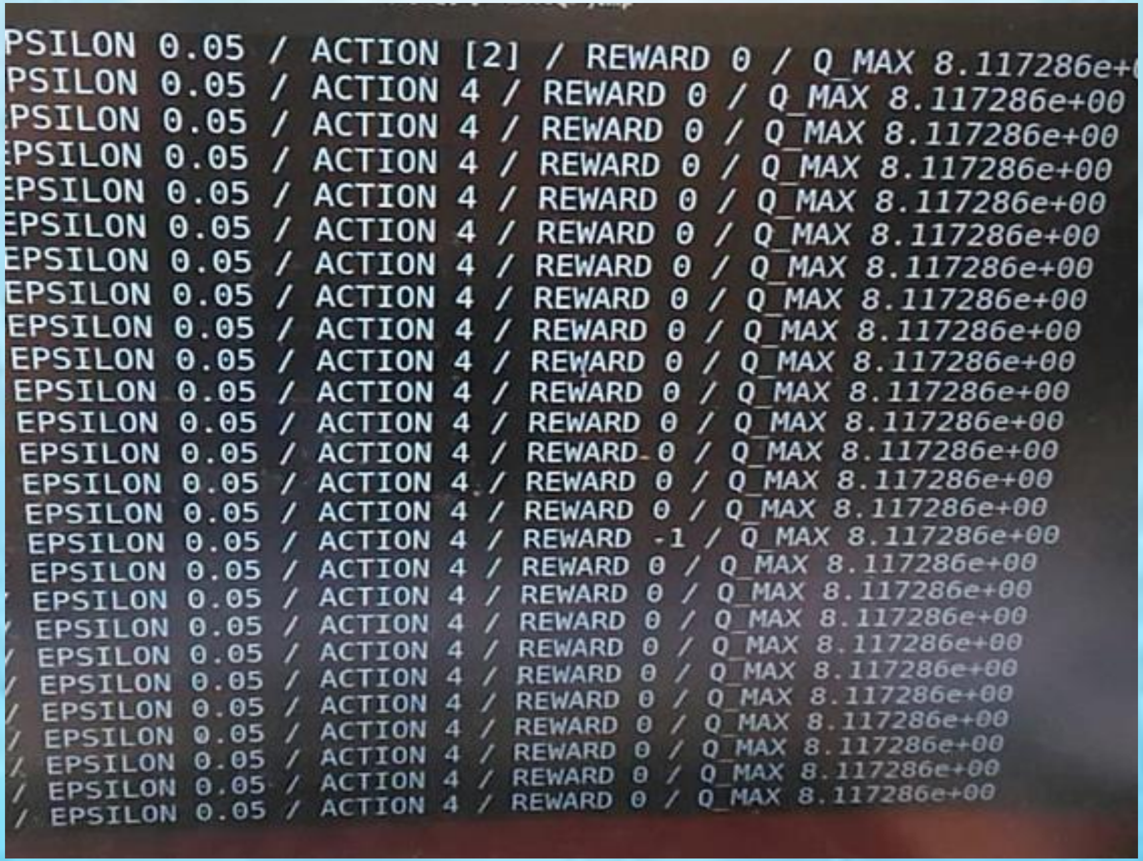
explore:500000 e_s: 1 e_f:0.05

Solution

- Trying different reward with respect to particular action
  - We tune the reward of get point and lose point
  - The rule base agent has the same speed as the ball, so it must be smash to get point.
  - So we expect that when the reward of getting point is much higher than losing point, the player will keep smashing.

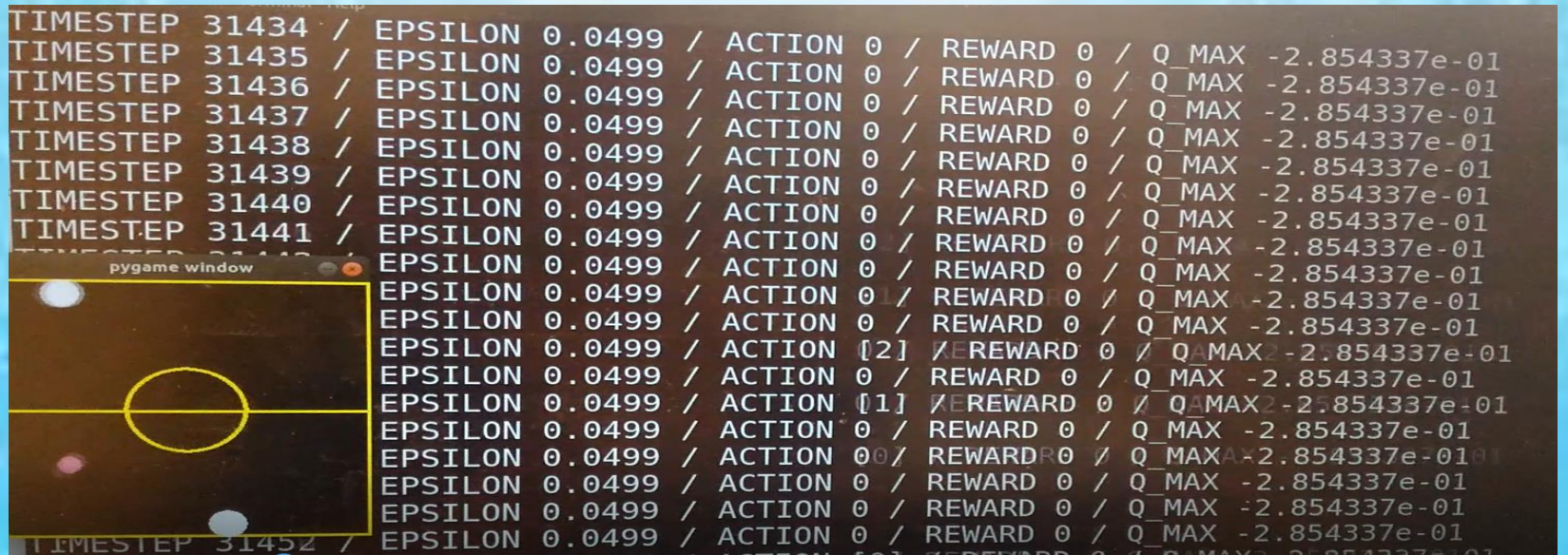
Result

- In our game “ACTION 4” means smash. This is the result of high reward for getting point than losing point.



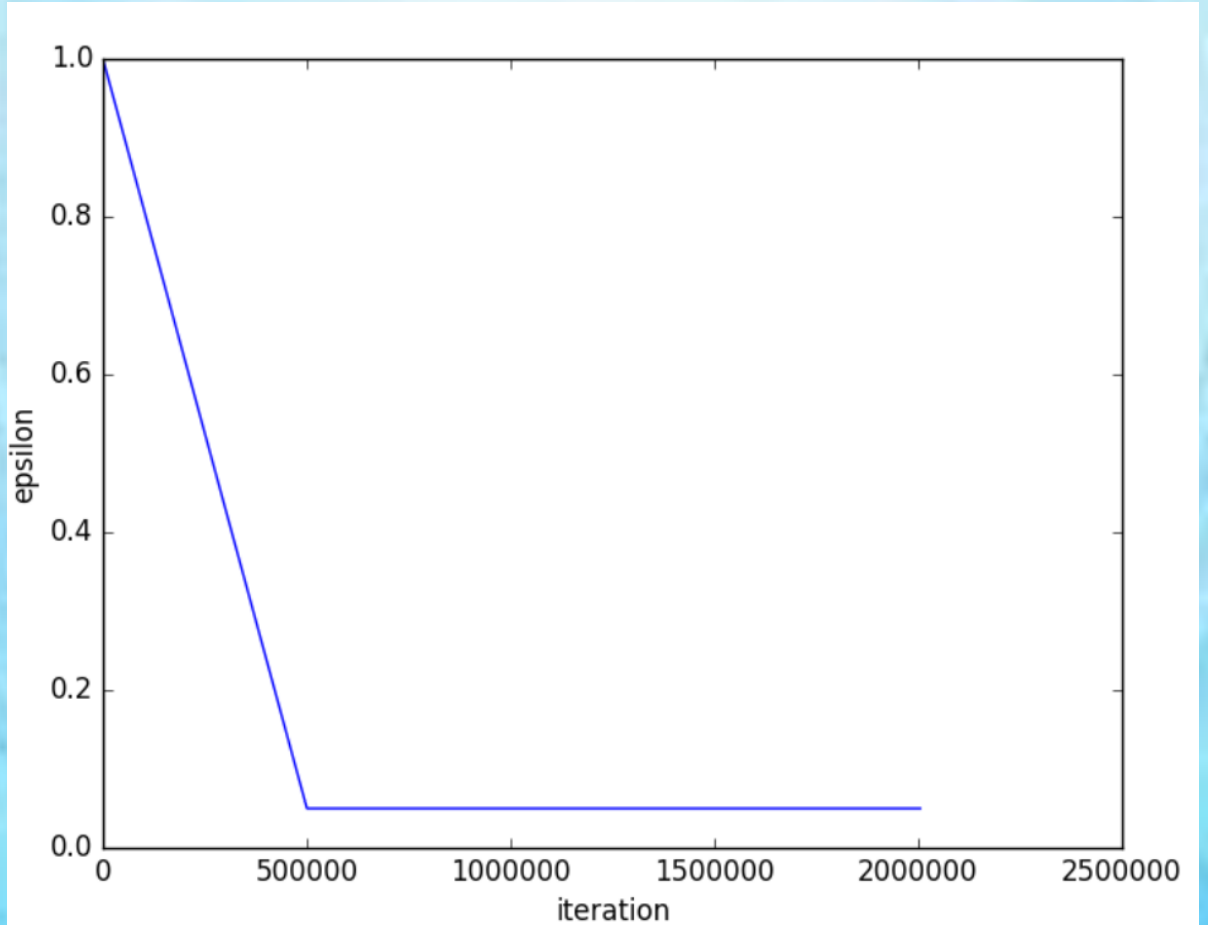
Result

- CNN use pixels of the input screenshot to compute output
- We transform the screenshot into binary image (black & white), and it make the CNN fail to recognize the position of the ball and become blind.
- ACTION 0 means “stay” in our game. The player can only stay at the same position with the failed training.



Result

- The linear annealing exploring rate with respect to iterations



Reference

- inspiration:
  - https://www.youtube.com/watch?v=iArc6FTLISA
  - https://gym.openai.com/
- tool used:
  - https://github.com/dennybritz/reinforcement-learning
  - https://github.com/asrivat1/DeepLearningVideoGames?fbclid=IwAR3DqNdsRTUSWdMROAx-O19IXwQd1CXxDyYL3KnSKrQEz8uUC1Qm9mgnRrM