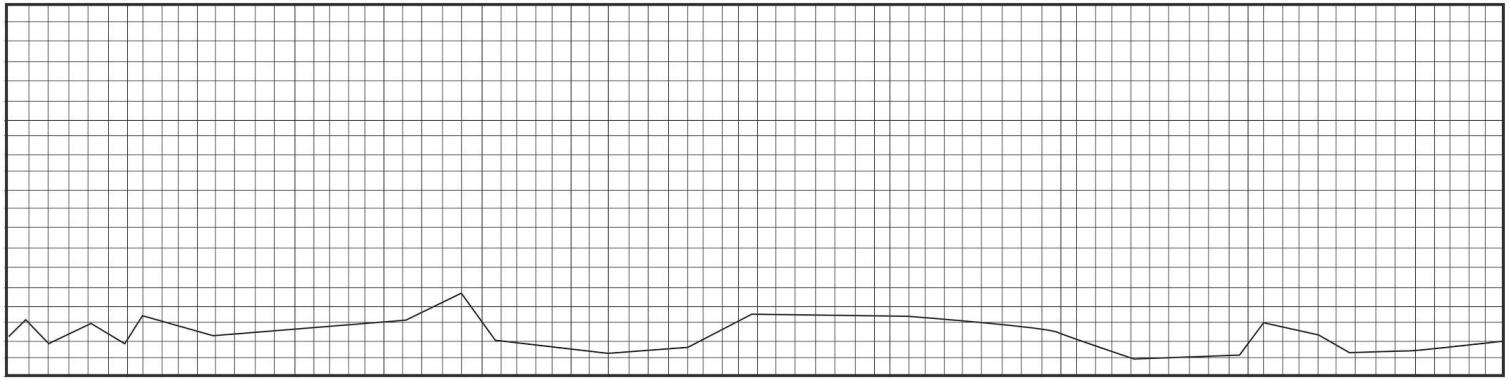
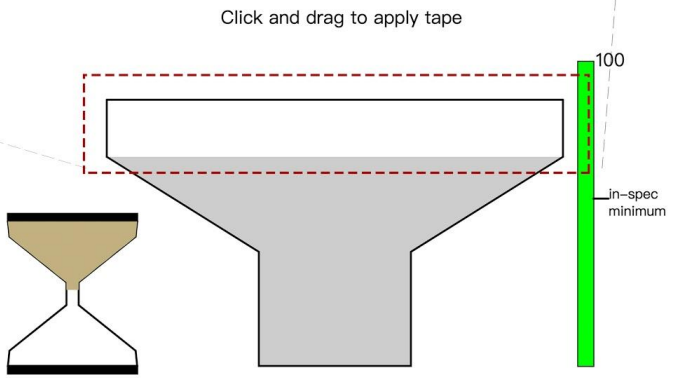
# Simulation of process of thermal spray operation

## Based on Deep Q network

Jiaoting Zhang  Ruoyi Yang  Wei Guo  Zhipeng Xu

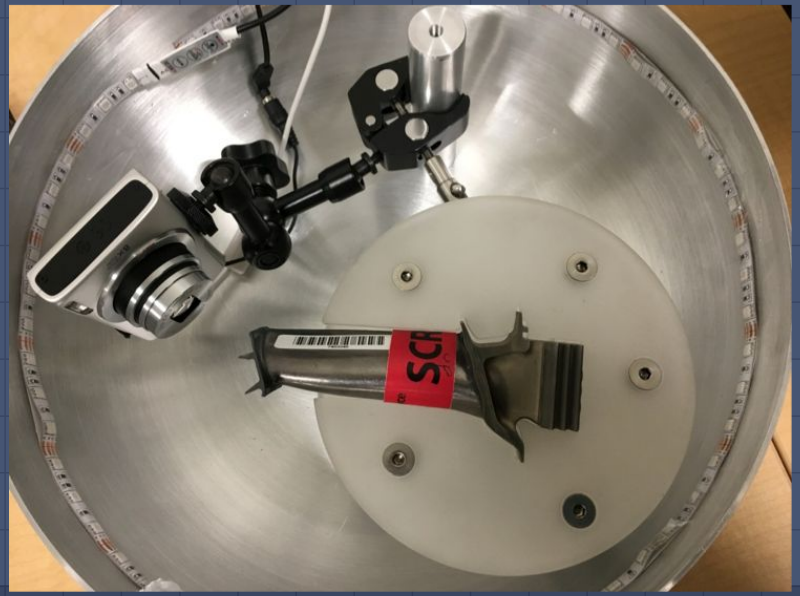
600 * 75 to 60*75





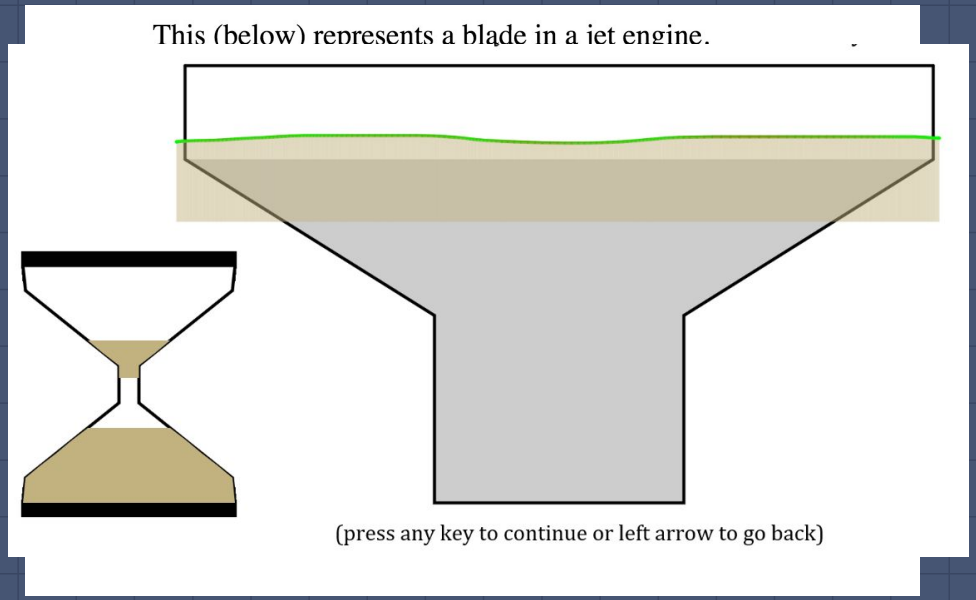
# Introduction of masking game

# Background



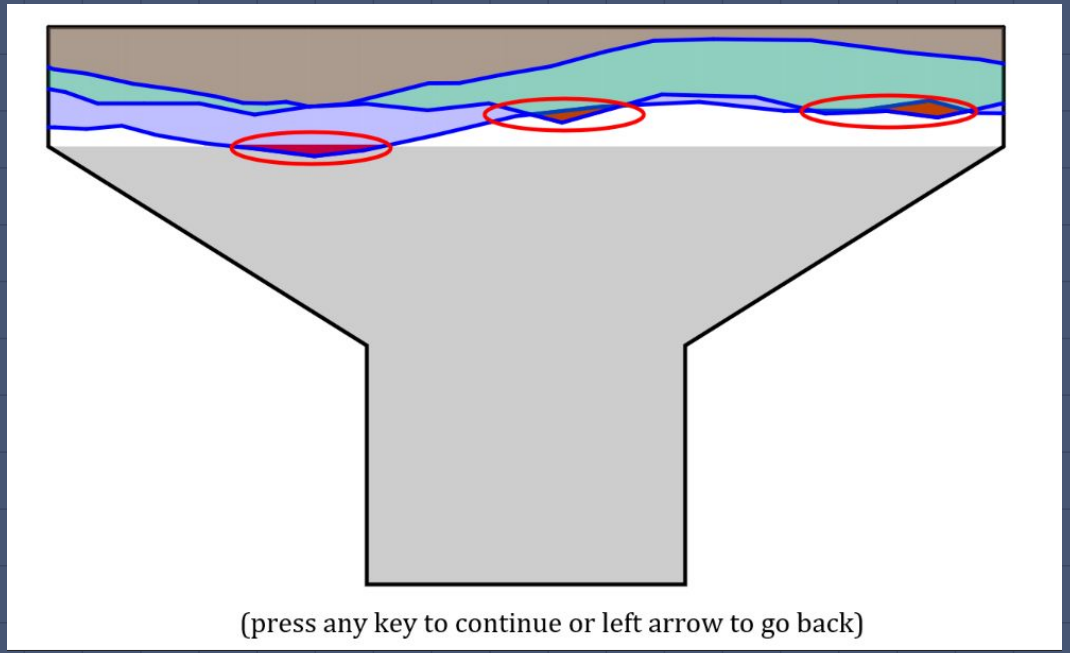
- Thermal spray operation still heavily rely on manual labor.
- Our project focus on the masking process in thermal spray operation combined the deep q network.
- Goal: The thermal spray can mask the blade without human interference.

# Game



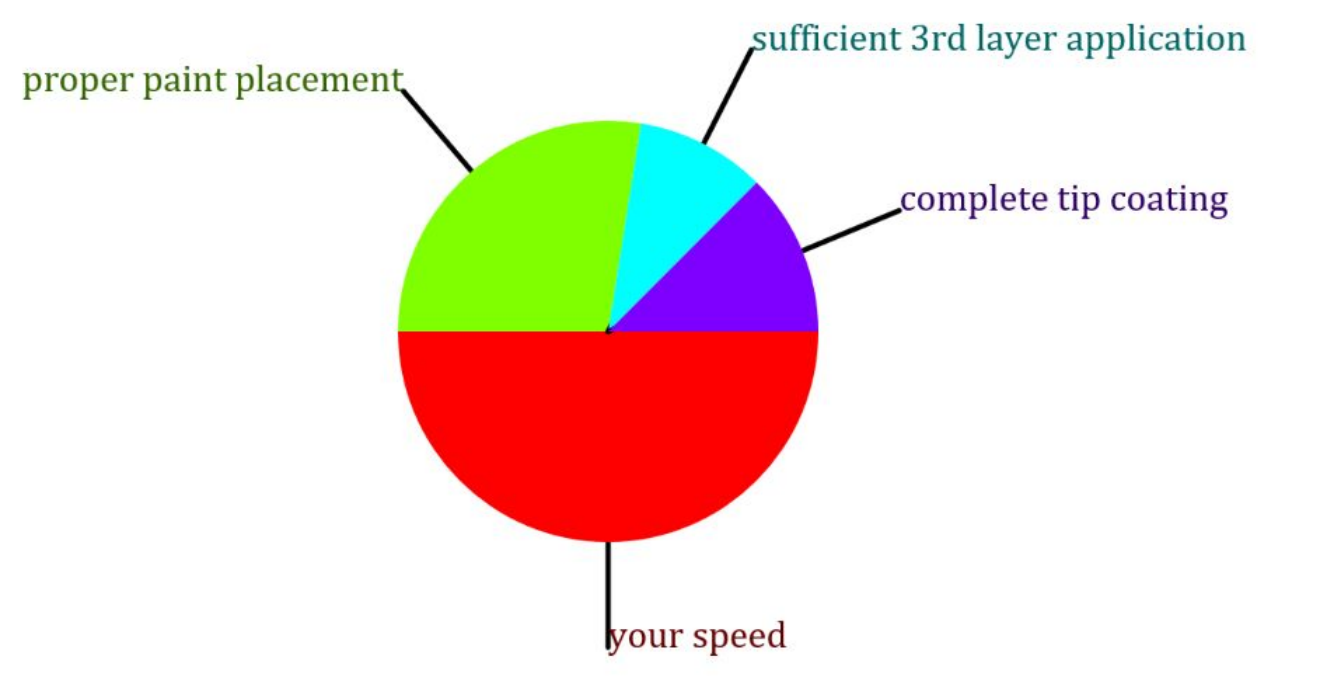

- We use the picture to represent a blade in a jet engine and this blade needs to be coated in 4 layers with 3 tapes.
- When applying the tape, we can use the mouse to control the direction.

# Example:




- The layer are applied correctly.
- The layer are applied incorrectly.

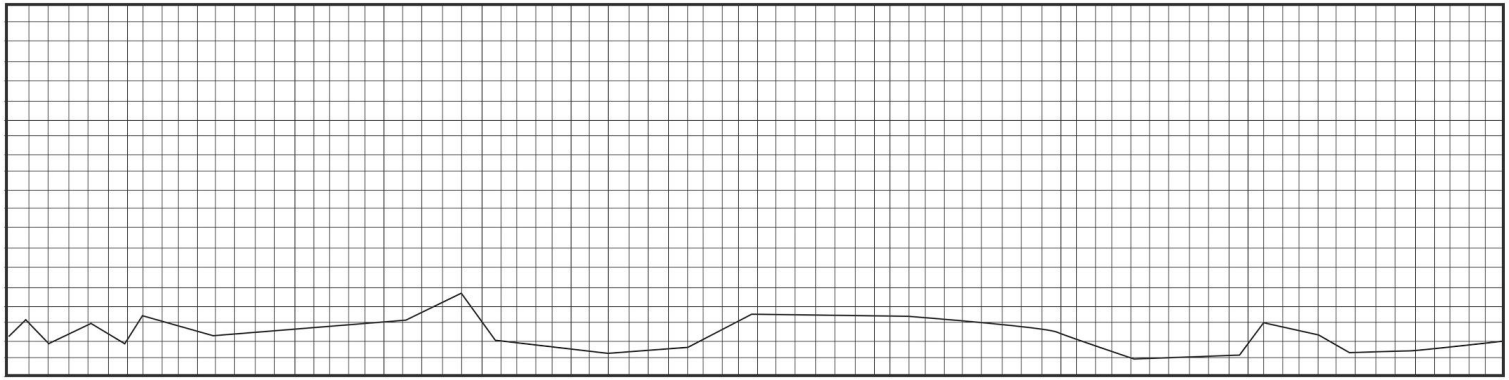
# Score is determined by:

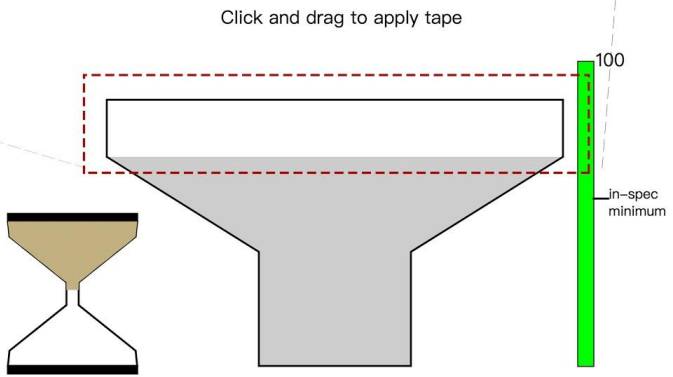


# Environment

- The game is based on website, interaction of agent and game will be achieved manually.
- After agent finish drawing a layer, game will generate random noise on the paint. What agent draws is not counted into reward function of game.
- We use mouse to simulate the behavior of agent on the website. Some necessary information can be reached after one layer is finished.

600 * 75 to 60*75





# State space and action space

- **State space:** there are 61 possible x-coordinates and 76 possible y-coordinates. Since there are three layers of painting, total state space will be 61 x 76 x 3 = 13908
- **Action space:** 76 possible y-coordinates
- **Transition :** While not reach end of layer, agent will seek next y coordinate, x coordinate adds up 10. Mouse will simulate the movement. While reach end of layer, action decide the beginning point of next layer, mouse will not simulate the movement.

# Deep Q Neural Network

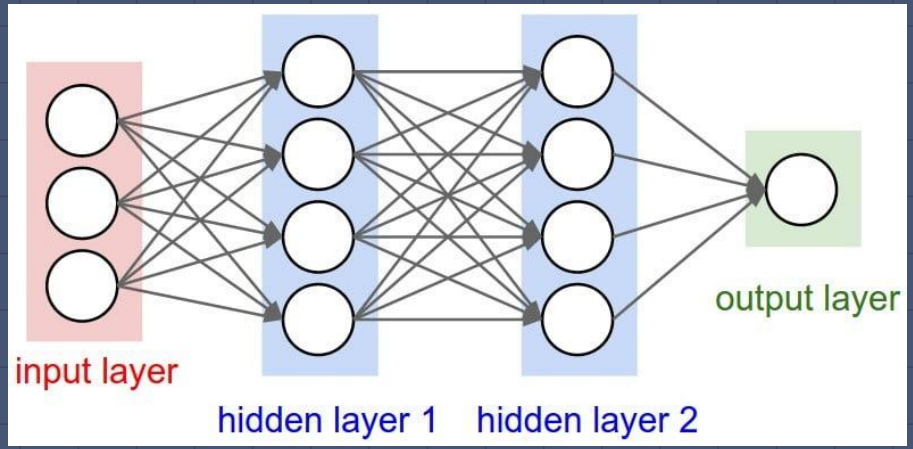
# Introduction of Deep Q Neural Network

- Since, We have 13908 states and and in each state, we have 76 actions. The Q-table is unrealistic.
- So, we use Neural Network here work as a function approximator for Q value.
- The cost function here is the mean square error of the prediction Q value and the target Q value.

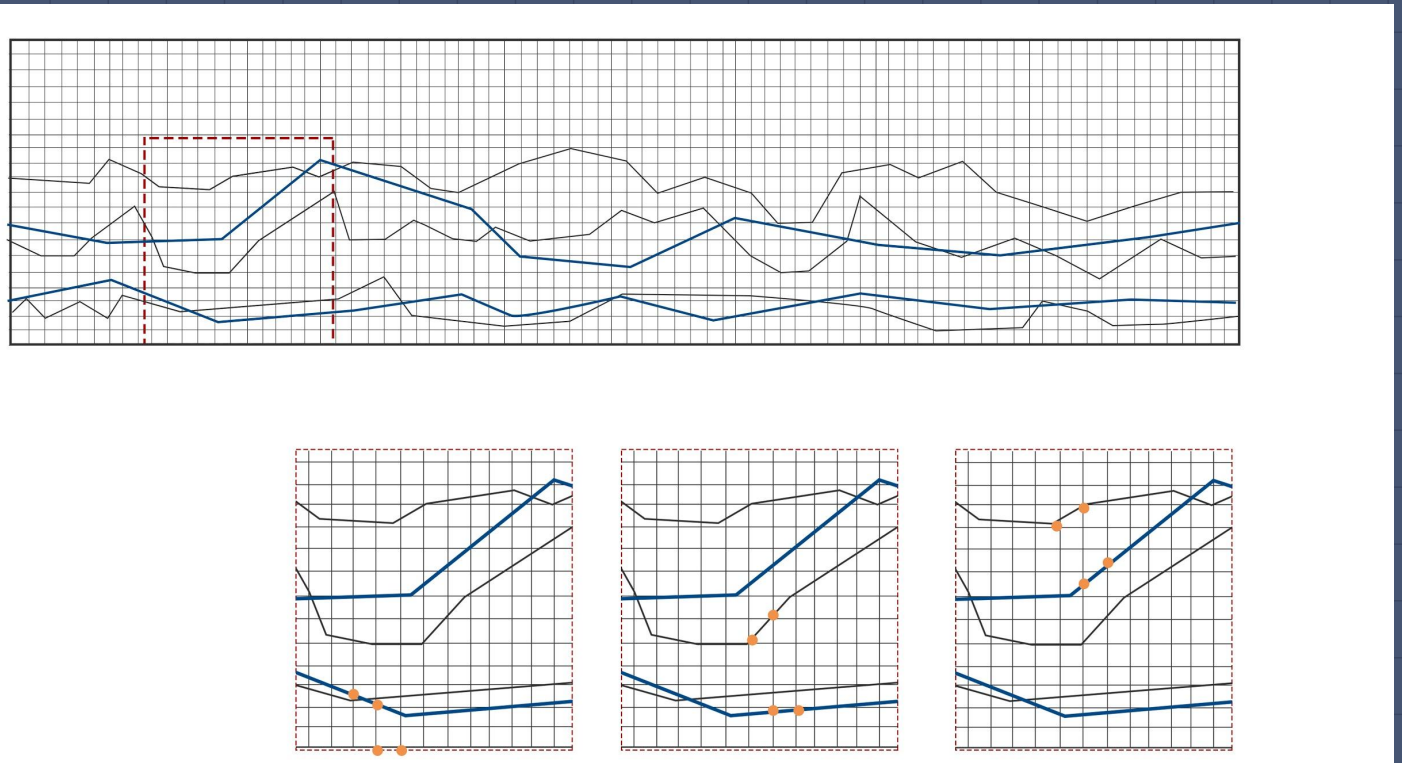
$$Loss = \frac{1}{2}.[\underbrace{r + max_{a_{t+1}}(Q(s_{t+a}, a_{t+1}; \theta_{t-1}))}_{\text{target}} - \underbrace{Q(s, a; \theta)}_{\text{prediction}}]^2$$

# Introduction of our model

- Two Hidden Layer Neural Network
- Input: 4 y-coordinate
- Hidden layer 1 : 32
- Hidden layer 2 : 64
- Output: 76



# Input of our network : (based on the line with noise)



# DQN Algorithm

**Algorithm 1** Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
**for** episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  **for** $t = 1, T$ **do**
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  **end for**
**end for**

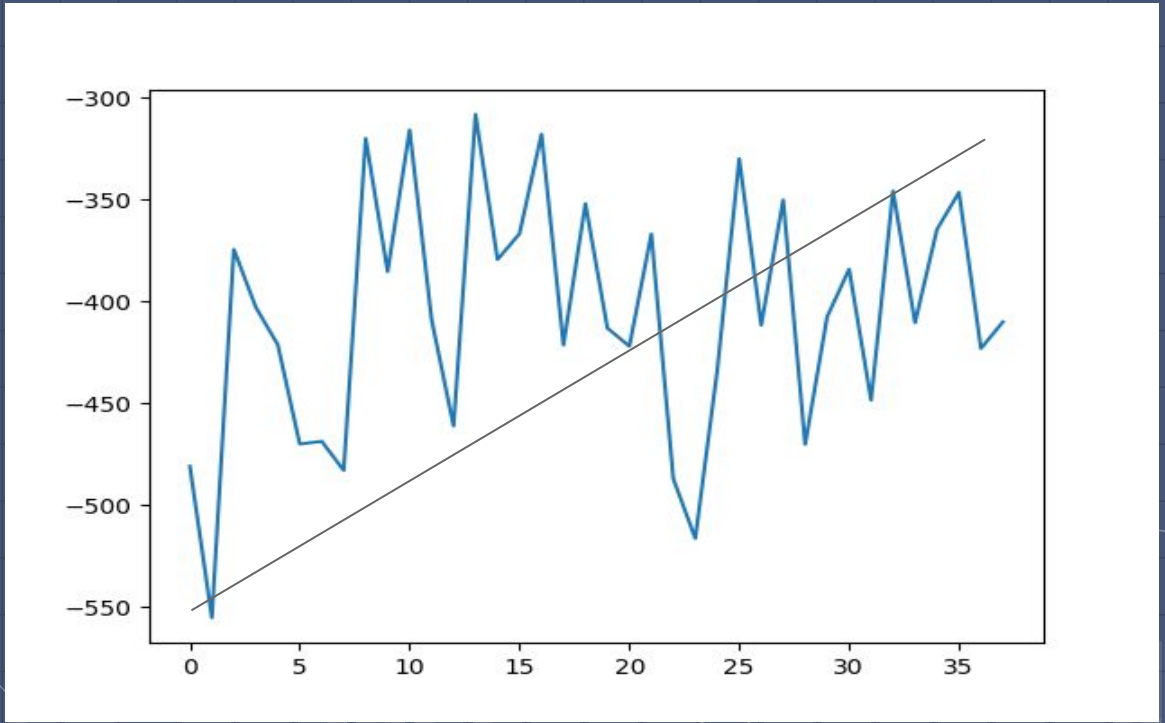
# Results & Further work

# Results

- Lines get locally optimized but not globally optimized.
  - After first 100 times, three lines will draw horizontal line on same y-coordinate.
  - After 1,000 times, two lines separate.
- In training mode, the line have trend to draw above its previous line.
- The grades get better in training.

# Training Results

- 40 grades in training steps
- The training results are to be shown in video.



# Further Work

- Training the model for more times (about 100,000 times)
- Use Neural Network with more hidden layer
- Optimize the web masking-game environment
  - movement of mouse
  - interaction between web environment and training model

Thank you!
Any questions?