Minecraft Maze Runner

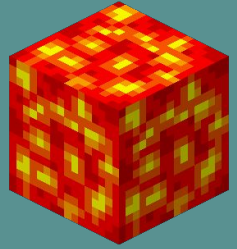
By Team 12

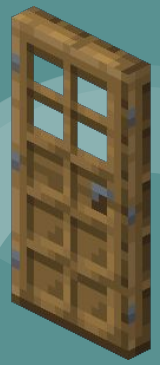
Group Members: Xiaolei Jiang
Yuxiang Qian
Jeffrey Eric Su

Project Website: https://yuxiangq777.github.io/
Github Repo: https://github.com/yuxiangq777/175project.git

Abstract

- Project goal: Training agent to find diamond block in different level mazes
- Easy level mazes: solved by Tabular Q learning
- Medium level mazes: solved by deep Q learning
- Hard level mazes: not solved yet...






Introduction

- Environment: Mazes in Minecraft
  Easy level mazes: 5 by 12 mazes with random generated lava
  Medium level mazes: 5 by 12 mazes with random generated lava and slimes
  Hard level mazes: 12 by 12 mazes with random generated lava, slimes, and doors

- Approach: Reinforcement learning based on states

- Algorithms: Tabular Q learning is good enough for lava only but has bad performance when we added zombies. Therefore, we switched to Deep Q learning instead
- https://www.youtube.com/watch?v=19KZK7K41z4

Background

- Tabular Q Learning

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

$$Action = \begin{cases} \max_a Q(s, a), & R > \varepsilon \\ Random\ a, & R \leq \varepsilon \end{cases}$$

R is random number between 0 and 1, ε is the exploration factor between 0 and 1. If ε is 0.1, then 10% of the times, the algorithm will select a random action to explore corresponding rewards.

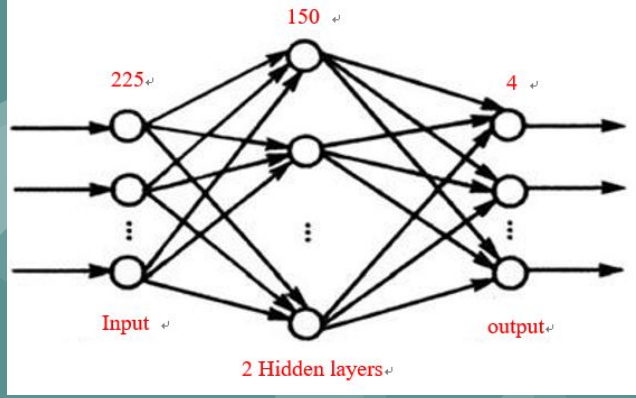
- PyTorch for Deep Q Learning

Problem Statement

- Maze Solving with random obstacles
- Agent can see a 3v3 block square around it
- Agent gains +500 for reaching lapis block, +2000 for diamond block, -1000 for dying
- Lapis blocks serve as sub goals to help agent reach final goal (diamond block)

Method

- Tabular Q learning
- Deep Q learning
  - 3*3 blocks of observation
  - 5 types of blocks: gold block(wall), lava, stone, diamond block, lapis block
  - replay buffer to store the previous 4 states
  - Feed-forward Neural Network
    - Input layer with 3*3*5*(1+4)=225 nodes
    - 2 hidden layer each with 150 nodes
    - Output layer with 4 nodes
  - Learning rate
  - Epsilon
  - Discount factor

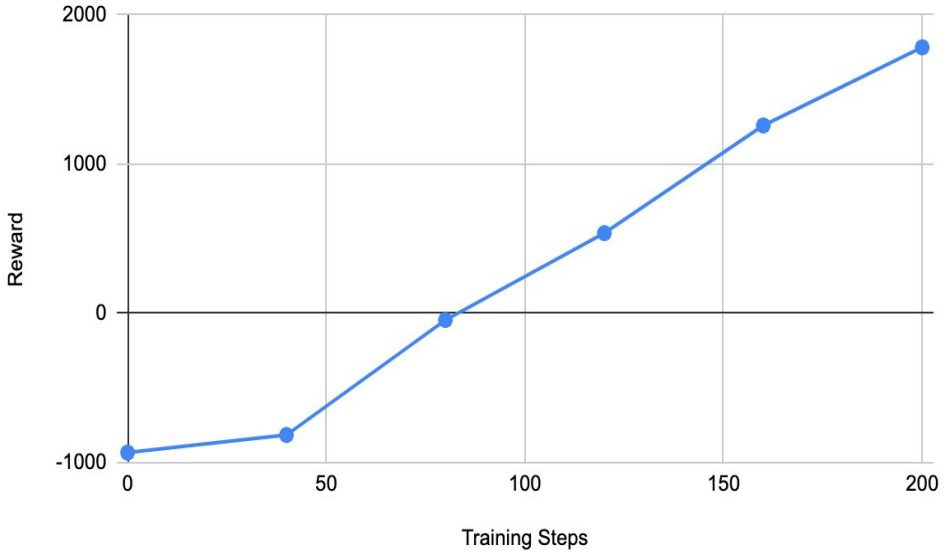


Experiments

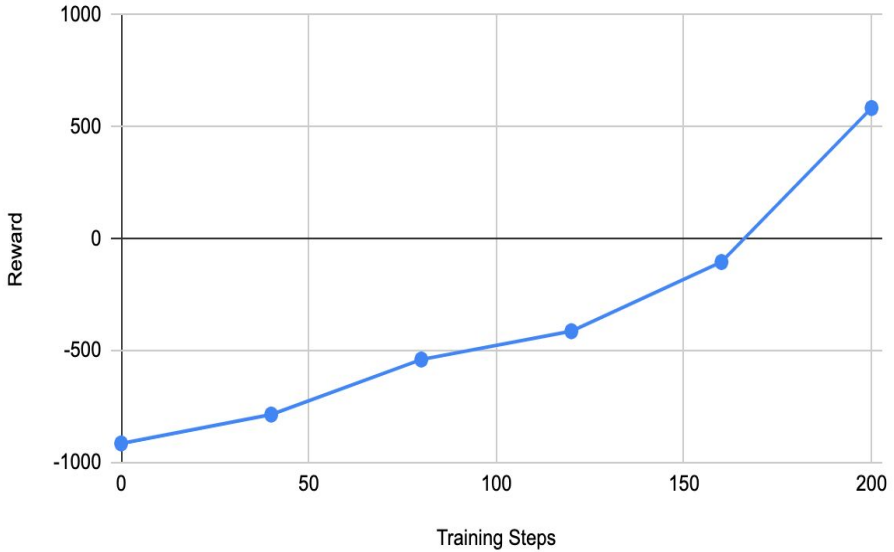
- Tuning hyperparameters
- Learning rate
  - Too high → stuck in sub-optimal solution
- Epsilon
  - Too high → explored the environment too randomly in the beginning and easily fell into the lava

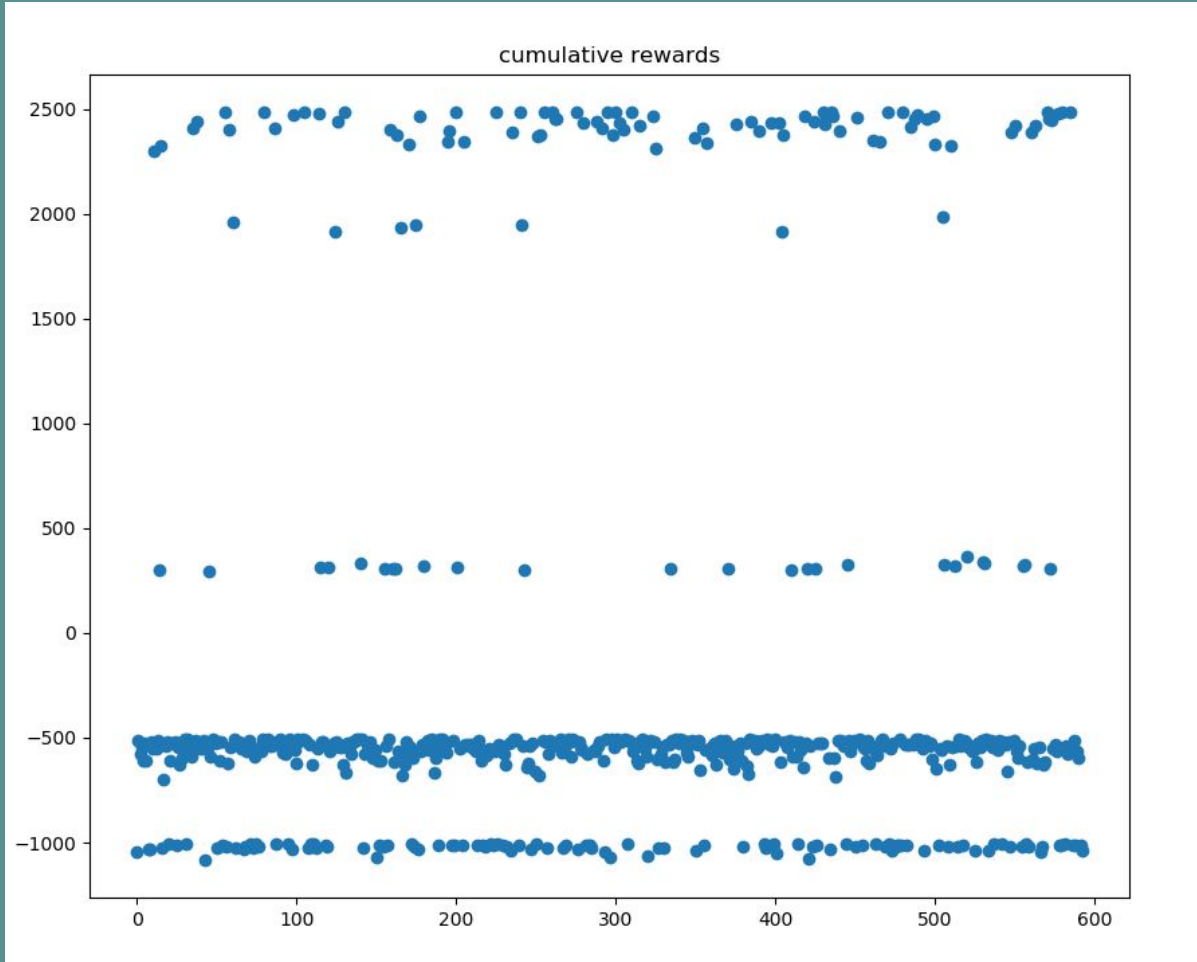
Results

Tabular Q Learning Average Rewards on Same Easy Maze



Tabular Q Learning Average Rewards on Different Easy Maze




cumulative rewards
2500
2000
1500
1000
500
0
−500
−1000
0
100
200
300
400
500
600

Reference

1. Microsoft's Malmo tabular q-learning example (https://github.com/microsoft/malmo)

2. Pytorch documentation (https://pytorch.org/)

3. Hui, Jonathan. "RL — DQN Deep Q-network" July 16, 2018. https://medium.com/@jonathan_hui/rldqn-deep-q-network-e207751f7ae4

4. Deep Q neural network example codes https://github.com/dannym08/DungeonMasters/blob/master/deep q learning.py

Questions?