MINESWEEPER REINFORCEMENT LEARNING WEB APPLICATION

Daniel Short

PROJECT OVERVIEW

- Components
  - Custom Minesweeper Environment
  - Reinforcement Learning (Deep Q-Network) Models
  - Web-based Interactive Application
- Tech Stack
  - JavaScript
  - CSS
  - Python
    - PyTorch
    - Gym
    - Flask
  - Visual Studio

WHAT IS MINESWEEPER?

- **Objective**: Click on all non-mine cells without clicking on a mine.
- **Starting the Game**: Click any cell to begin (first click is safe).
- **Numbers**: Numbers indicate how many mines touch that cell (including diagonally). No number means it isn’t touching any mines.
- **Flagging Mines**: Right-click cells that have mines to mark them (*optional*).
- **Using Logic**: Use numbers and flagged cells to logically deduce safe cells.
- **Winning the Game**: Click on all safe cells.
- **Avoiding Mistakes**: Clicking on a mine ends the game immediately.

METHODOLOGY

- Environment & Data
  - Dynamically generated Minesweeper boards
  - First cell click clears a 3x3 area
  - Between 10% and 20% of cells are mines
  - Grid sizes from 5x5 to 10x10
  - Good Example: https://minesweeperonline.com/
  - Bad Example: https://minesweeper.online/game/4524282468

METHODOLOGY

- Reinforcement Learning Approach
  - Deep Q-Networks (DQN)
  - CNN for Board States
  - Replay buffer
  - Epsilon-greedy policy
  - Curriculum learning strategy

TRAINING AND RESULTS

- Training Setup
  - Network
    - Deep Q-Network (DQN)
    - Double DQN
    - Dueling DQN
  - Board State
    - Max Pooling
    - Adaptive Pooling
    - Global Pooling
  - Memory Replay Buffer
    - Standard Replay Buffer
    - Prioritized Replay Buffer

TRAINING AND RESULTS

- Hyperparameters
  - Episodes = 10,000
  - Batch size = 64
  - Train Frequency = 50 episodes
  - Update Target Model Frequency = 1,000 episodes
  - Success Threshold = 50%
- Performance Metrics
  - Success Rate

DQN MODEL POINT VALUES

- Positive Rewards
  - Open a cell correctly: +0.3 points
  - Forced guess (correct): +0.3 points
  - Forced guess (incorrect): +0.3 points
  - Win the game: +1.0 points
- Negative Rewards
  - Guess (correct): -0.3 points
  - Guess (incorrect): -0.3 points
  - Lose the game: -1.0 points

DEMO OF THE WEB APPLICATION

- Interactive Web Interface
  - AI Gameplay
    - Visual gameplay of the RL agent
  - User Gameplay
    - User-selectable grid sizes

CHALLENGES AND SOLUTIONS

- Challenges
  - Balancing exploration vs. exploitation
  - Managing computation complexity
  - Finding point values to incentivize learning
  - Determining what is determined successful performance in training
- Solutions
  - Improved epsilon decay strategy
  - Curriculum learning for stability
  - Trial and error

LEARNING OUTCOMES

- Learning Outcomes
  - Experience with reinforcement learning (big topic with ChatGPT recently)
  - Custom environment creation
  - Model Deployment and UI designing

CONCLUSION AND FUTURE WORK

- Project Results
  - Double DQN outperformed
  - Adaptive Pooling outperformed
  - Standard Replay Buffer outperformed
- Project Achievements
  - Successful RL agent training
  - Effective curriculum learning implementation
  - Interactive and user-friendly web application
- Future Directions
  - Implement Visual Transformers instead of CNNs
  - Expand to similar games like Nonogram
  - Optimize model parameters