Mini Project
Chess AI Agent

Kaustubh Kale

Team Intro

- Kaustubh, MEng student with DA concentration

Problem Description

- Problem: Build an AI Agent that optimizes chess moves
- Simple rules, difficult gameplay (more states than atoms in universe)
- Board positions can be evaluated
- Requires complicated strategies, risk management

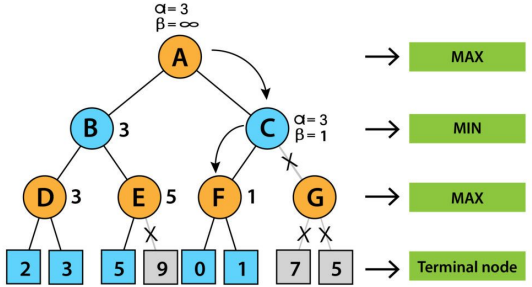
Approaches

- Traditional Search and Heuristics
- Minimax, Alpha-Beta Pruning
- Enhancements: Quiescence Search, Selectivity

- Reinforcement Learning
- Monte-Carlo Tree Search
- Q-learning

Alpha-Beta Pruning/Minimax

- Maximizer, Minimizer
- Requires Heuristic
- Enhancements (horizon effect)
  - Transposition Table
  - Quiescence Search
  - Selectivity
    - Botvinnik-Markoff Extension
    - Capture/Check Extensions
    - Mate Threat Extensions
    - Delta Pruning

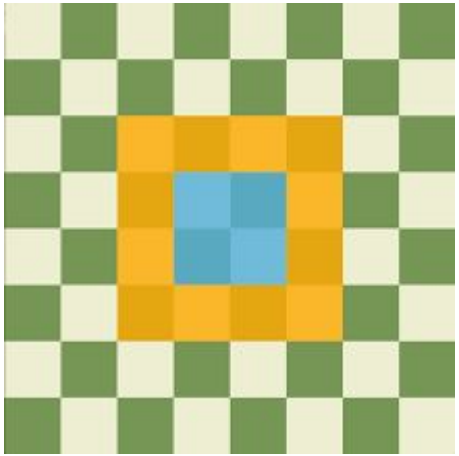


Reinforcement Learning

- Q-learning with engine playing itself
- Reward given by pieces
- Play until end of game or 100 moves

Heuristic

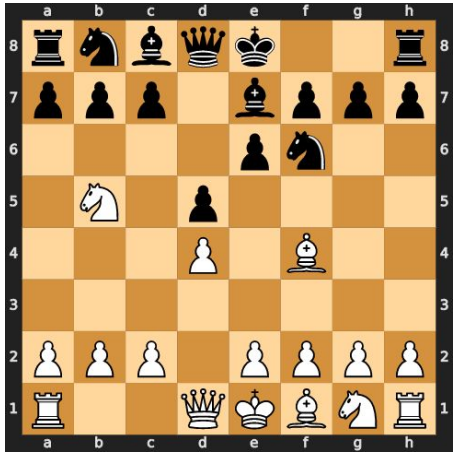
- Piece value (Pawn: 1, Knight/Bishop: 3, Rook: 5, Queen: 9)
- Mobility
- Centrality
- King Safety
- Pawn Structure
- Game Phase

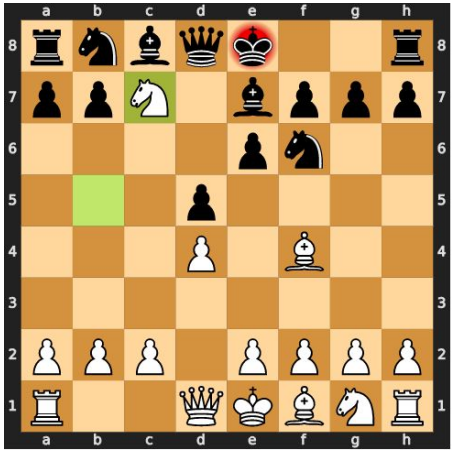


Results

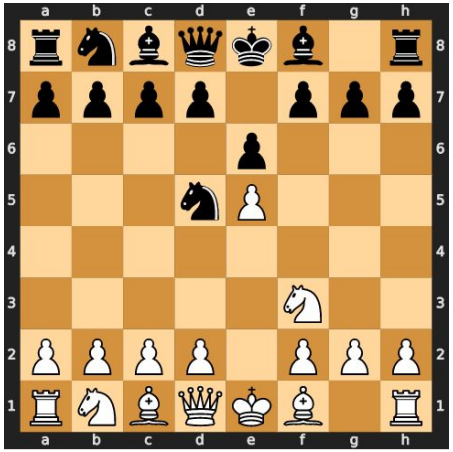
- Q-Learning agent moves make no sense
- Alpha-Beta agent shows great promise
- Estimated ~1000-1200 ELO in opening/middlegame phases (should outplay a beginner)
- Struggles heavily in the endgame due to long term plans

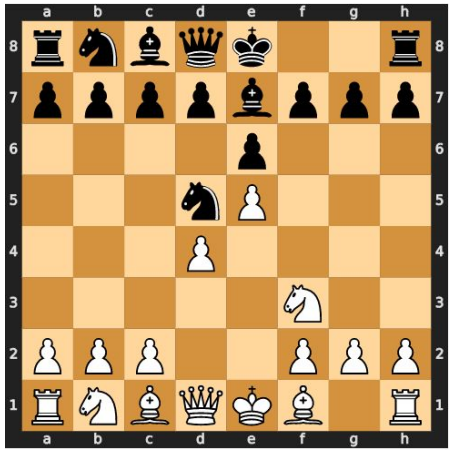
Winning Pieces

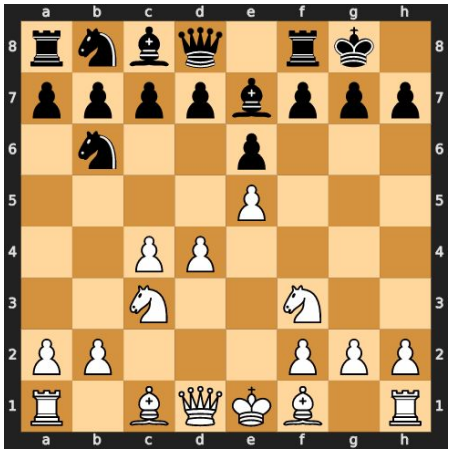




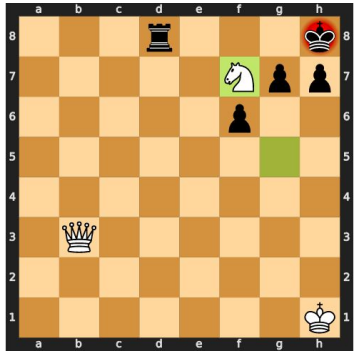
Centrality/King Safety

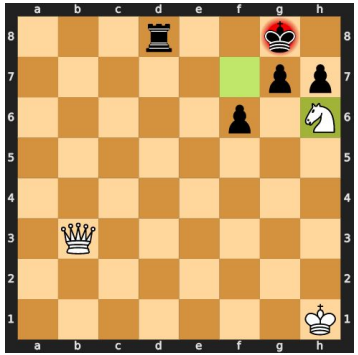


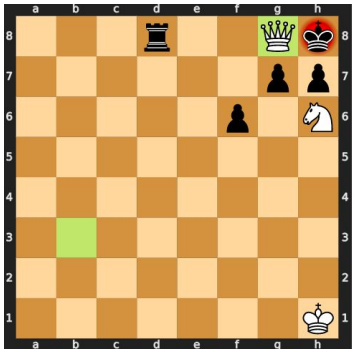


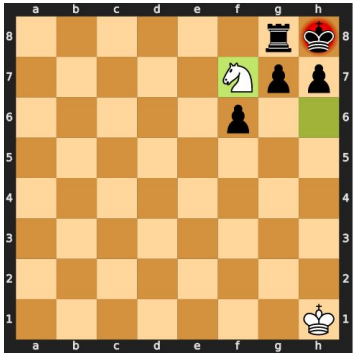


Checkmate

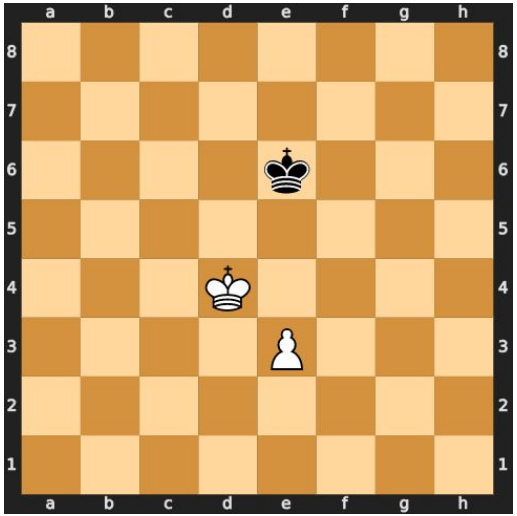


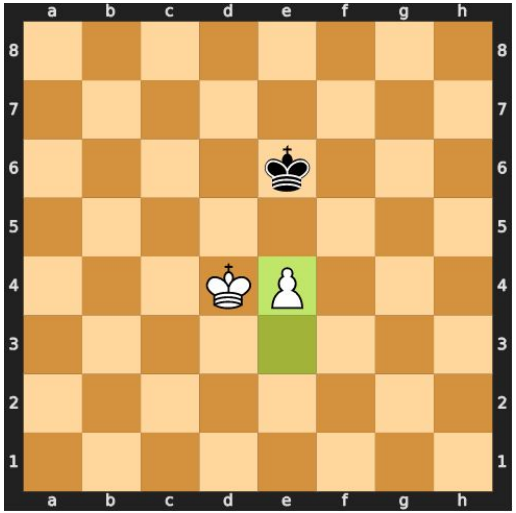






Endgame Struggles





Lessons Learned

- Creating an intermediate level chess agent is doable
- Heuristic = bulk of work
- Witty optimizations for search
- RL Agent’s require lots of training
- Board representation/hashing techniques

Future Work

- Parallelism
- Improving Heuristic
- Endgame Search
- Monte Carlo RL