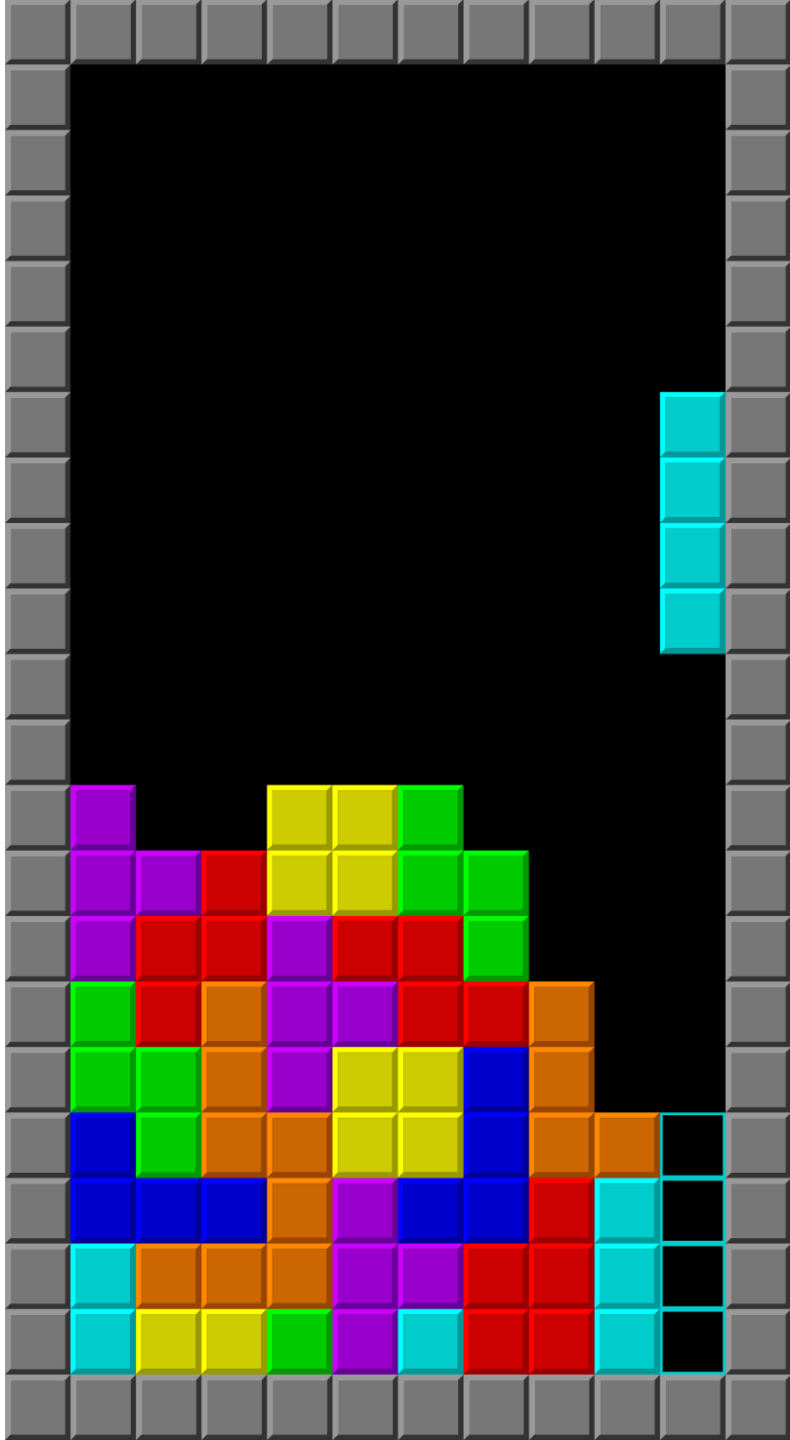
Deep-Q Learning Tetris AI

Ijahmin Balser

Problem

- Design an AI to score high in Tetris
- Tetris has a lot of possible states
  - Representation of game state can make this number even larger
- Smaller state space will lead to faster training and better performance for AI
- Common Tetris strategies tend towards long term pay-offs
  - Well designed reward function can help with short-term pay-offs

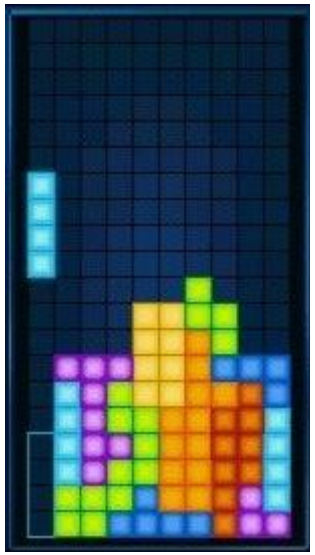


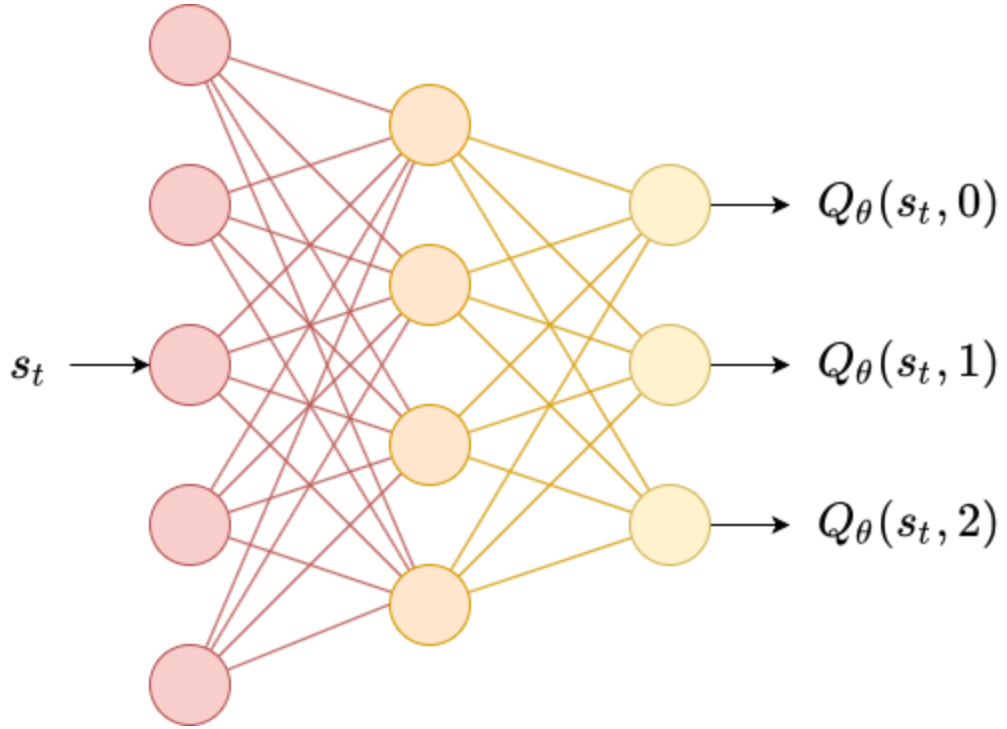
Use-Case Scenarios

- Reference for new Tetris players
- AI opponent in multiplayer Tetris games
- DQN Agent is transferrable other games/applications
  - Bug testing possibilities
  - Performance evaluation

AI Algorithm and Model

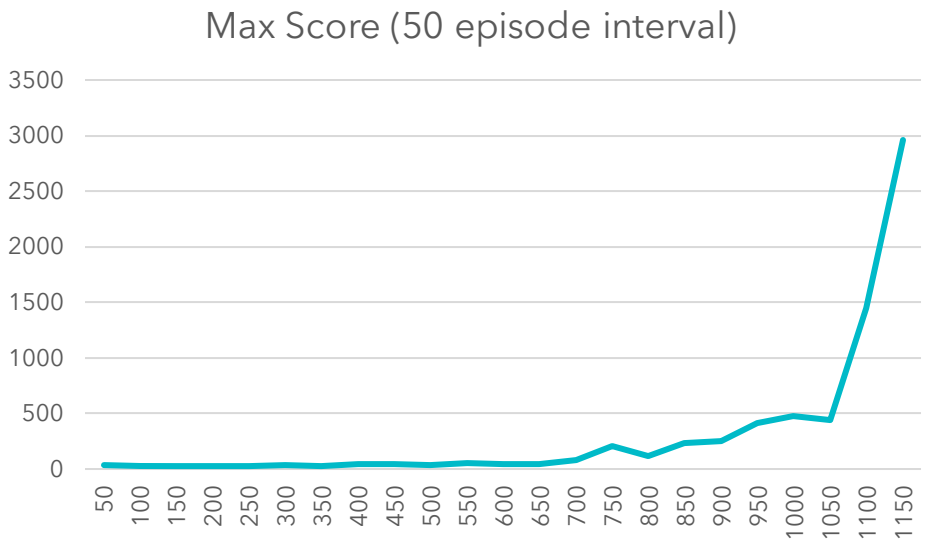
- Reinforcement Learning
  - Standard Q-Learning Algorithm
  - Deep Q-Learning Agent
- Game state evaluated based a few factors
  - Lines cleared
  - Number of holes
  - Bumpiness
  - Total height



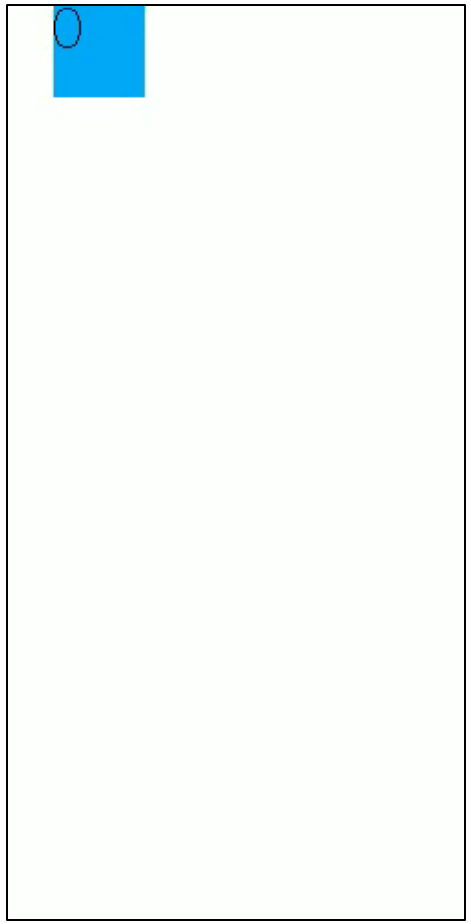


Results

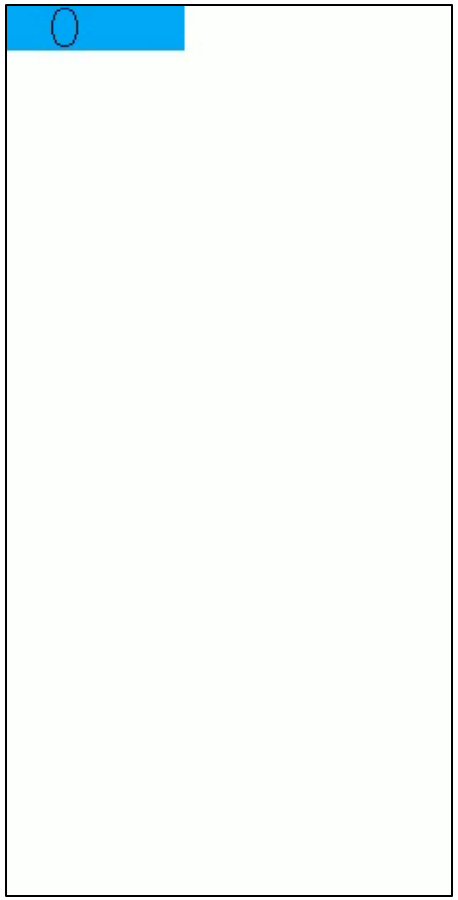
- Performance is heavily dependent on reward function
- Q-Learning policy produces little change when compared to a random policy
- Deep Q-Learning Agent improved rapidly
  - Episodes length becomes too long to record towards end of training set



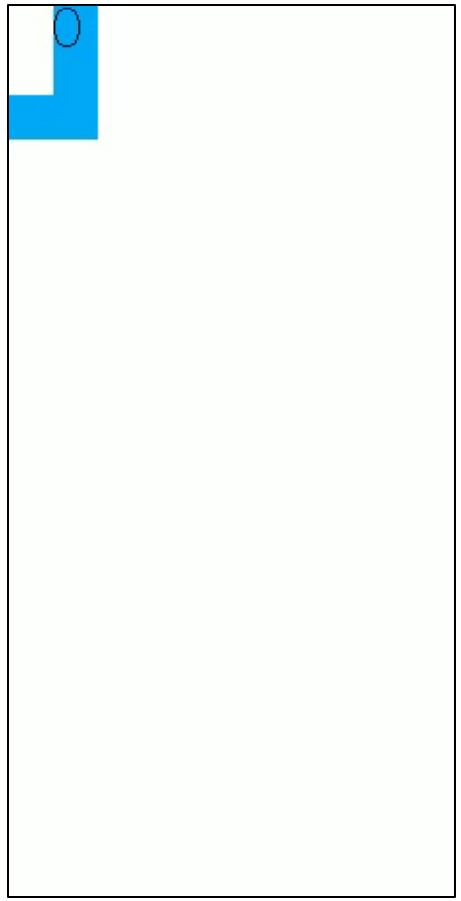
Demo



Episode 510



Episode 1180



Episode 1630

Lessons Learned

- Minimizing representation of game leads to a much easier time training
- Relying on exclusively long-term payoffs is not reliable
- Optimized reward function is one of the most important factors

References

- https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/
- https://github.com/nuno-faria/tetris-ai
- https://pytorch.org/docs/stable/tensorboard.html
- https://keras.io