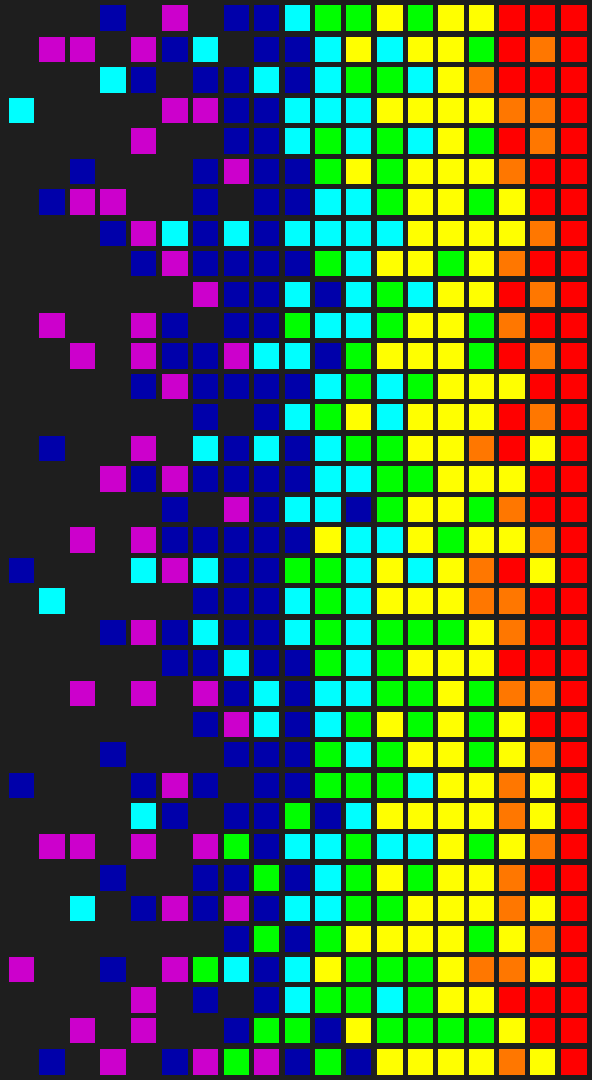
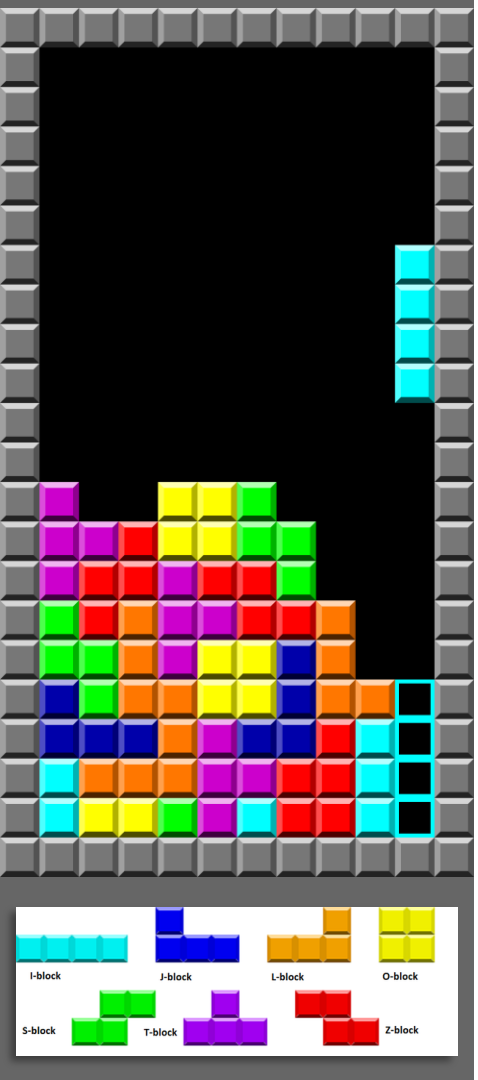
TETRIMAX

CS 4100 Final Project

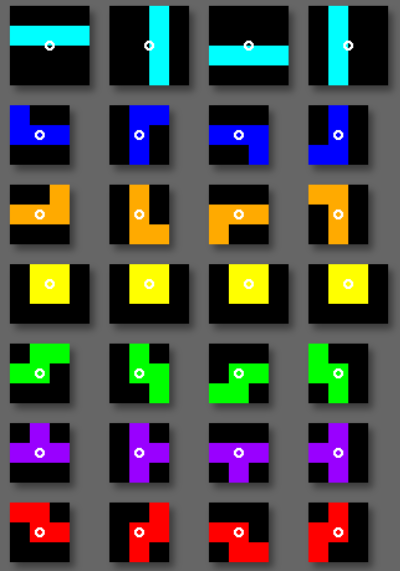
Connor Nelson, Mara Hubelbank, Tyler Passerine

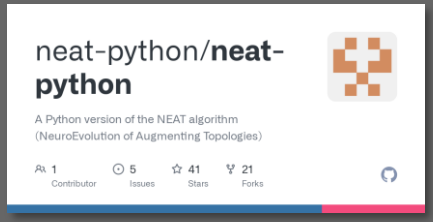




Problem Formulation

- **Project Goal:** Create agents to play **Tetris**
- **Game Objective:** Clear lines, ideally scoring "Tetrises" (4 lines at once)
- **Simplification:** Agents play **asynchronously,** focusing only on choosing the drop action which will result in the best new board.
- **Agent Tasks:**
  1. Take in the current game piece.
  2. Generate all possible drop moves + boards.
  3. Evaluate each board.
  4. Select the move with the highest board rating.



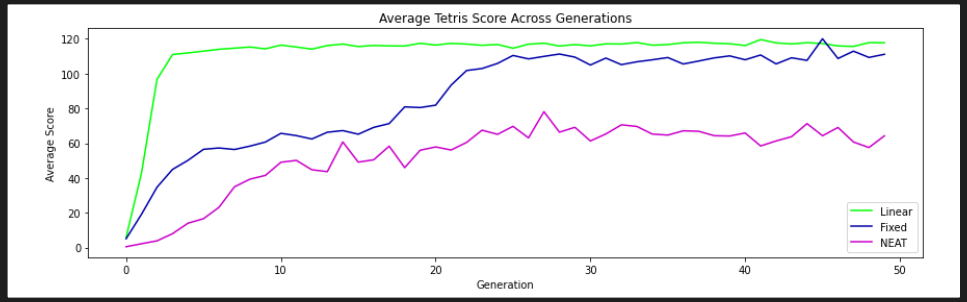


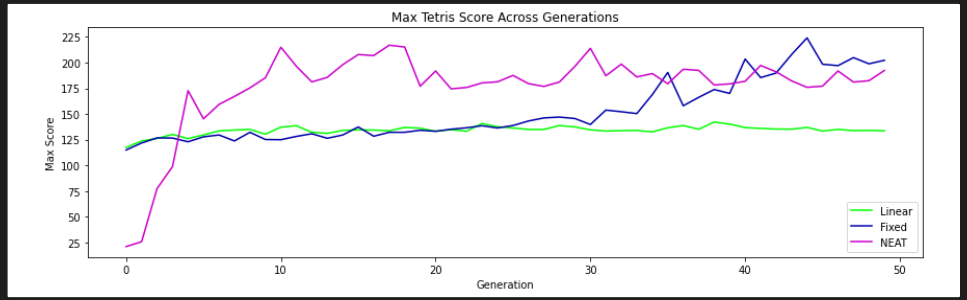
Algorithm Design

- 1st: **Expectimax** (state space explosion)
- 2nd: Genetic Algorithm Factory → **Agents**
- **Agents** are generated through **natural selection.**
- Agent "Factories":
  - **Linear:** Agents contain as many weights as features. The output of a given board state is given by $x_1w_1 + x_2w_2 + \ \ldots + x_nw_n$.
  - **Fixed-size NN:** Contains two hidden layers both with as many perceptrons as features.
  - NEAT-Python

Performance Analysis

- The less-complex **linear agent** starts off much higher due to population randomness.
- The complex **nonlinear agents** consistently achieve a higher **max score**, because they learn to combine features and score Tetrises.
- Which nonlinear agent is “best” depends on the metric of evaluation.





Conclusion & Reflection

- Agents' **goals** significantly impacted how they played.
  - Maximizing **survival** vs. **score**
- By Tetris % metric, our agents performed worse than humans.
  - Feature-based representation → Incomplete view
  - Lack of memory → No piece foresight / predictive analysis
- **Project Extensions:**
  - Independent, unsupervised agent learning
  - Advanced non-drop moves (ex. T-spins) to mimic human play

Thank you!

please like and subscribe