Economics & Computation

Connect 4

Reinforcement Learning

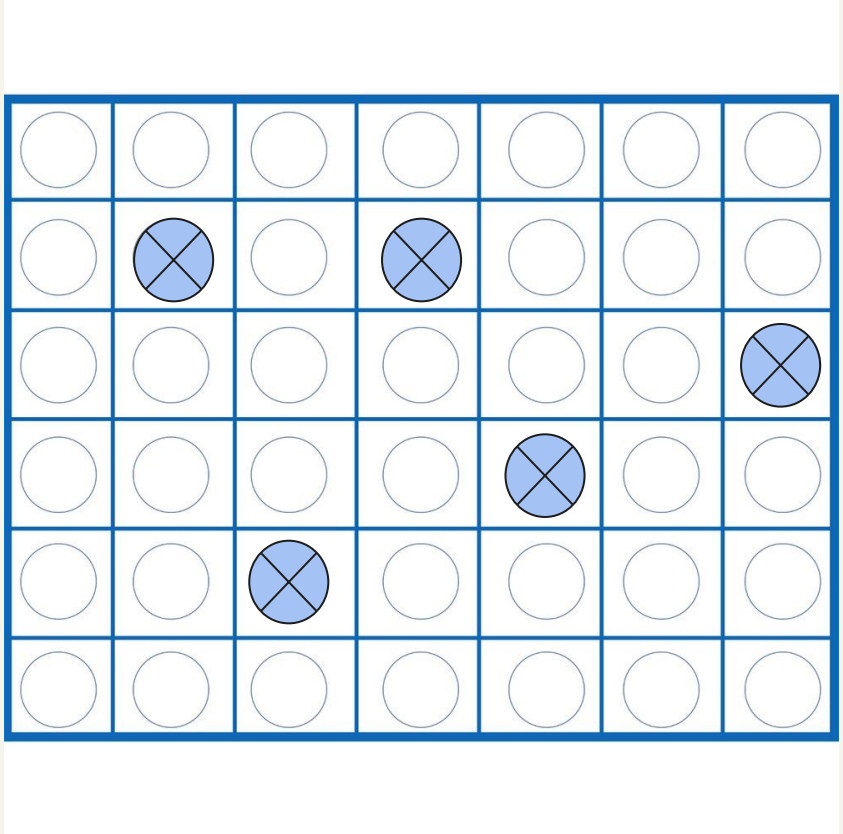
Venora Furtado, Tejaswi Tripathi

Rules of Connect 4

- 2 players, 7×6 board
- A player places a piece at the bottom of a column, and the pieces stack
- The first player to get 4 of their own pieces in a row (horizontally, vertically, or diagonally) wins
- Simple!

Our Goal

- Connect 4 is already a solved game. Current Minimax solutions exist that play this game robustly and can beat virtually any human player
- We introduce a new version in which certain cells are blocked: no piece can be placed there.
- Our goal is to build an RL bot that can beat the existing Minimax solution in this new environment.



What Has Been Done?

- **Minimax Algorithm with Alpha-Beta Pruning**
  - Commonly used in perfect-information games like Connect 4
  - Effective when combined with domain-specific heuristics
  - Allis (1994): Proved Connect 4 is a solved game—first player can force a win if he plays in the middle

- **Limitations of Classical Methods**
  - Assume full observability of game state
  - Poor performance in environments with uncertainty or hidden information

What Has Been Done?

Deep Reinforcement Learning

- **AlphaZero (DeepMind)**
  - Combines Monte Carlo Tree Search (MCTS) with deep convolutional neural networks
  - Trained via self-play; achieves superhuman performance in Chess, Go, Shogi
  - Designed for fully observable, fixed-rule environments

- **Limitations**
  - Performance drops in games with hidden or dynamically changing states
  - Inapplicable to modified Connect 4 with randomly occluded cells

What Has Been Done?

Imperfect Information Strategies

- **Libratus (Brown & Sandholm)**
  - Superhuman performance in no-limit Texas Hold’em
  - Uses counterfactual regret minimization and subgame re-solving

- **Limitations**
  - Tailored to betting mechanics
  - Not directly transferable to spatial board games like Connect 4

Why is This Important?

Motivation & Contribution

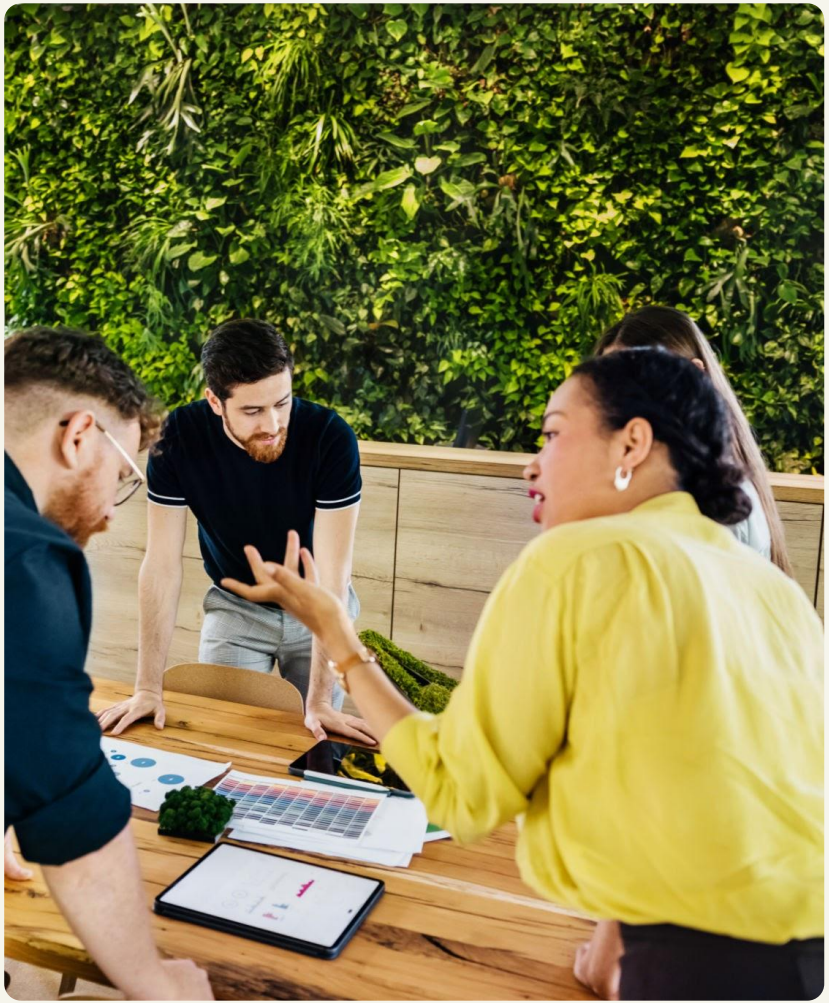
- **Current Challenges**
  - Existing models assume either full observability or stable rules
  - Dynamic uncertainty in our Connect 4 variant breaks these assumptions

- **Our Approach**
  - **Hybrid model combining:**
    - Adapted Minimax with probabilistic heuristics for hidden cells
    - Neural network trained via self-play for decision-making under uncertainty

- **Novelty**
  - No prior work combines symbolic search and neural representations for Connect 4 with hidden cells
  - Provides a new framework for studying strategy in partially observable environments

Why?

- Real-world decision making frequently involves uncertainty and partial observability—features not captured in standard Connect 4.
    - Financial trading, autonomous systems, recommender systems, and multiplayer games like poker
- The covered cells create a partially observable environment, forcing AI agents to infer possible board states and adapt their strategies dynamically.
- By comparing Minimax to RL, we aim to explore how different models cope with uncertainty in adversarial environments.



Minimax Agent

Approach:

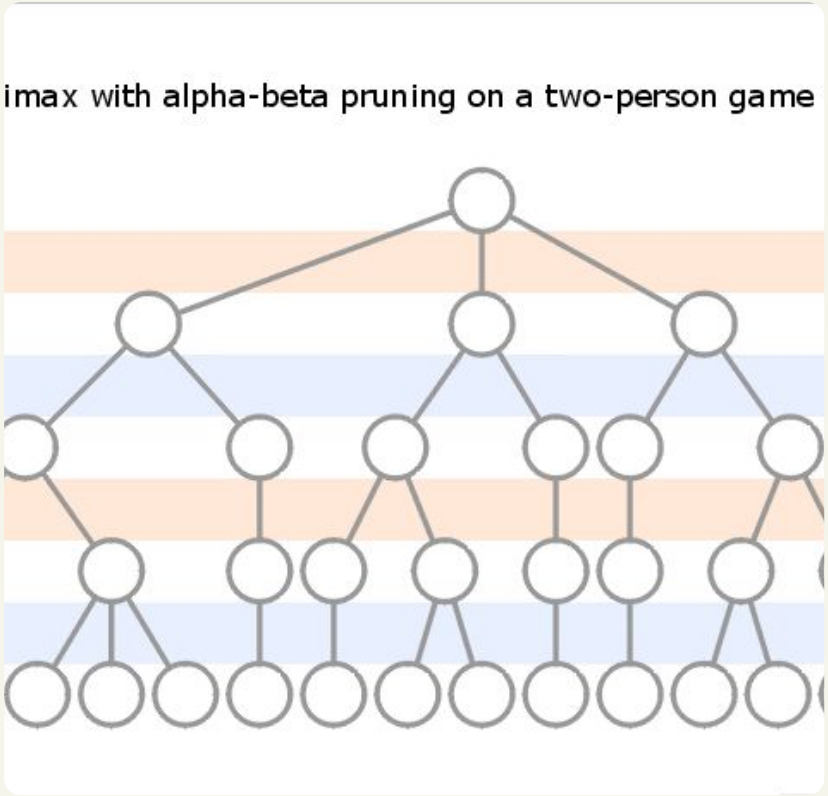
- Implements the Minimax algorithm with alpha-beta pruning to search the game tree.
- Evaluates potential moves up to a certain depth
- Picks the move with the best score based on:
  - Win detection
  - Center control
  - Piece height (closer to bottom = better)
  - Preventing opponent's wins

Strengths:

- Deterministic and smart — always picks the best move based on evaluation.
- No training required.

Limitations:

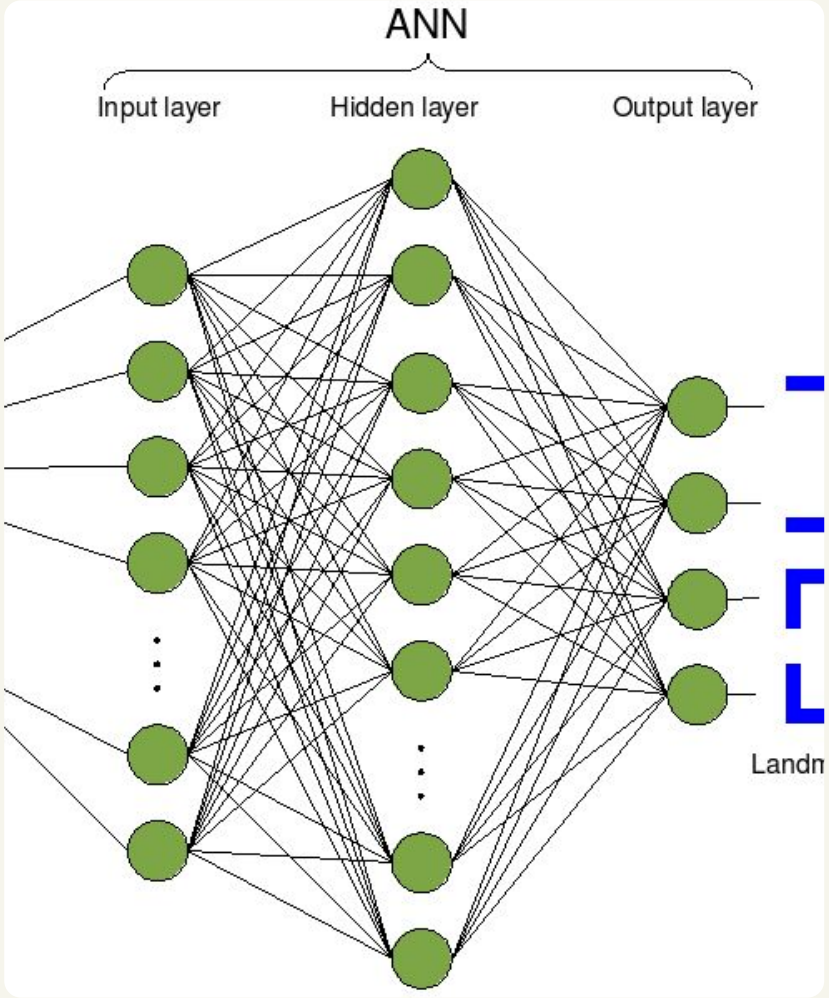
- Performance and speed degrade with deeper searches
- Model does not learn



Neural Net

Approach:

- Uses a 3-channel image-like encoding of the board:
  - Player 1 tokens
  - Player 2 tokens
  - Randomly hidden tokens (adds uncertainty)
- A Convolutional Neural Network (CNN) maps board states to move probabilities.
- Trained using REINFORCE, a policy gradient method:
  - Plays games against a random agent.
  - Learns from wins (+1), losses (-1), and draws (0).



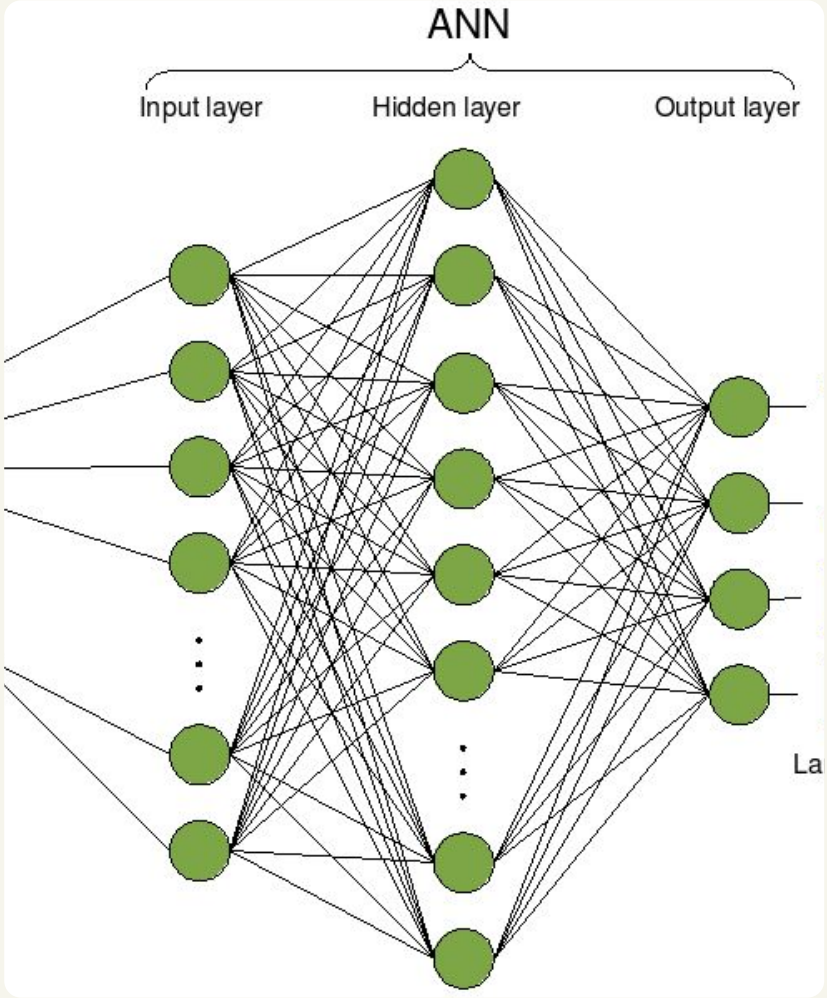
Neural Net

Strengths:

- Learns strategies over time.
- Generalizes well with enough training.

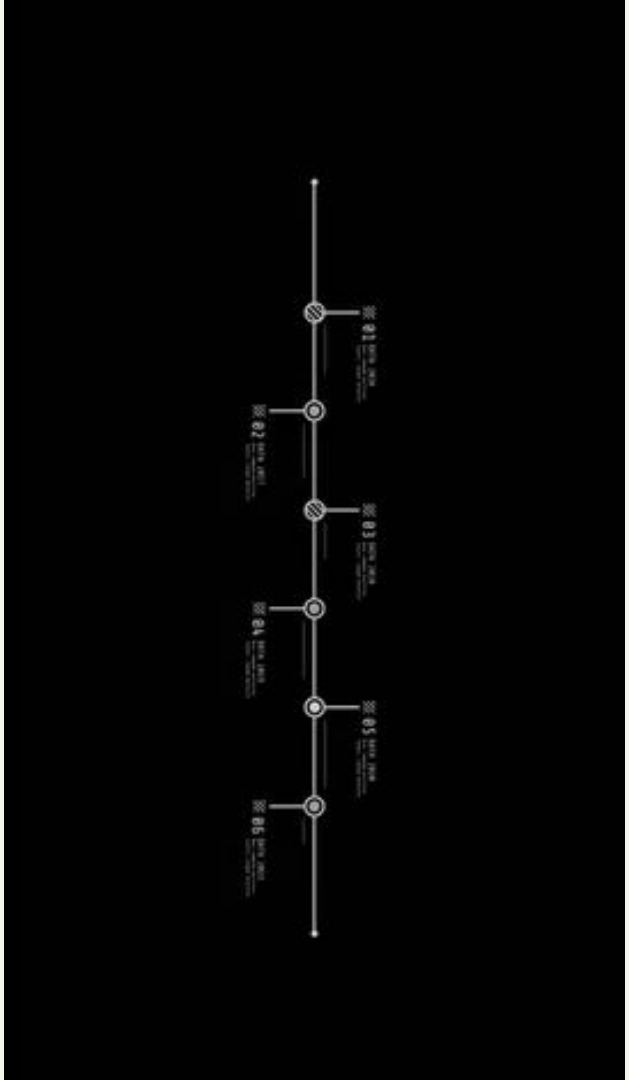
Limitations:

- Training is slow and reward signal is sparse.
- Performance depends heavily on architecture and training time.



Timeline:

- **Sprint 1:** Build core Connect 4 engine with masked cells
- **Sprint 2:** Implement basic Minimax with alpha-beta pruning
- **Sprint 3:** Modify Minimax to estimate hidden cell probabilities
- **Currently on Sprint 4:** Design and train NN model on partial states + Tune Minimax Algorithm
- **Sprint 5:** Run tournament between agents under varying hidden cell conditions and collect data, analyze win/loss outcomes, evaluate robustness
- **Sprint 6:** Compile results into final paper and presentation



Baseline Test

- We tweaked the Minimax algorithm to handle cases of blocked cells.
- We ran this Minimax solution against a random agent in an environment with 4 blocked cells and achieved a win rate of 80%.

```
Results for Minimax vs Random:
Total games: 5
Minimax wins: 4 (80.0%)
Random wins: 1 (20.0%)
Draws: 0 (0.0%)
Average move time for Minimax: 0.138s
Average move time for Random: 0.000s
```