Applying α-β Beam Search & CNNs to Checkers

Gio Cerutti

Checkers Problem Overview

- Checkers, chess, and other board games have been a focus of computational research for a long time
- Checkers was the first "classic game" played by a computer (1952)
- Checkers has a very large search space: $5*10^{20}$ possible positions
  - Solved in 2007 by Jonathan Schaeffer (18 years of CPU time spent on α-β search)

Possible Approaches

- Two approaches to game playing:
  - Type A: Consider all possible moves
    - Pros: Exact, always makes correct decisions
    - Cons: Slow, memory intensive
  - Type B: Ignore moves that seem bad
    - Pros: Quicker, reaches deeper than Type A using same amount of memory
    - Cons: Might accidentally throw away some of the best moves
- My method uses the Type B approach
  - α-β beam search with CNN evaluation

My Approach

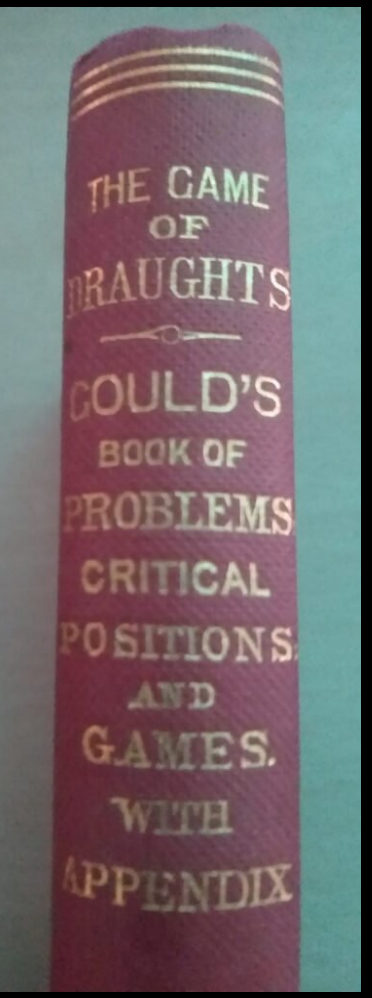
- α-β beam search with CNN evaluation
- Heuristic evaluation function trained using 12,892 puzzles found in checkers books, some over 100 years old

Convolutional Neural Net (CNN) Details

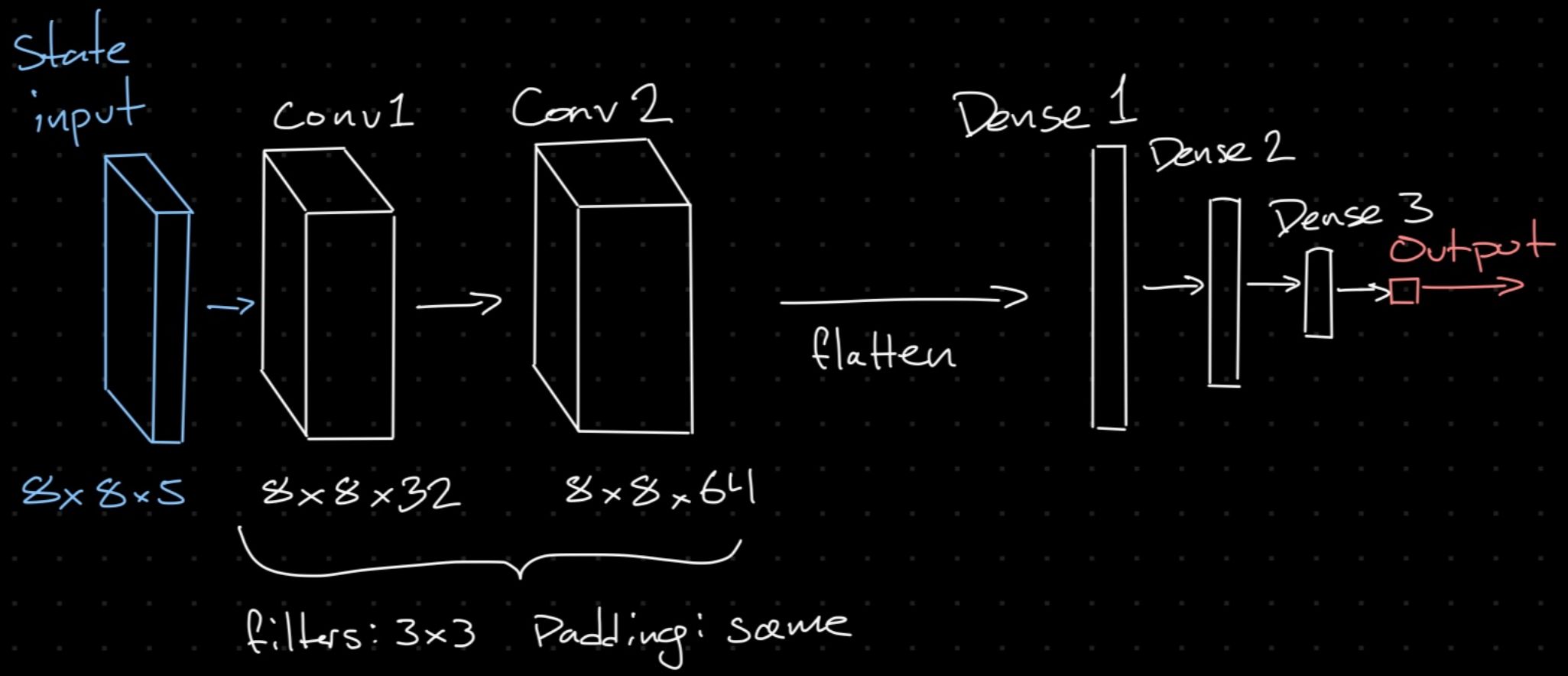
- Regression model (single output node)
- CNN input: 3D 8x8x5 grid that describes the state of the checkers board
  - Data included in each layer: Red pieces, red kings, black pieces, black kings, whose turn is it
- CNN output (training labels):
  - +1 if red is winning
  - -1 if black is winning
  - 0 if game is drawn

Training Data

- Almost 13,000 puzzles found in old checkers books:
  - *Tricks, Traps, and Shots of the Checkerboard*
  - *Beginner's Problems*
  - *Gould's Problem Book*
  - *Boland's Bridges*
  - *Let's Play Checkers*
  - *The Hand of D.E.O.*
  - *Midget Problems*
- Training labels: 1 if red wins, 0 if drawn, -1 if black wins
  - Intended to simulate the probability that either side will win in a certain position



CNN Architecture



Playing the Game

- Have you ever played checkers using a NumPy array?
  - Know your board coordinates
- The AI takes a long time to think, even with shallow depth and narrow beam
- Is the AI good at playing checkers?

```
[[0 2 0 2 0 2 0 2]
 [2 0 2 0 2 0 2 0]
 [0 0 0 0 0 2 0 2]
 [2 0 0 0 2 0 0 0]
 [0 0 0 1 0 0 0 0]
 [1 0 1 0 1 0 1 0]
 [0 1 0 0 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
Picking up:
Row: 5
Col: 2
Moving to:
Row: 4
Col: 1
```

Results

- Slow
- Bad at checkers
- How to improve:
  - Use a hardcoded evaluation function
  - Use reinforcement learning
  - Use a faster library than Keras
  - Use a faster machine than Google Colab