From GO To Draughts

Haozhe SUN & Tong ZHAO

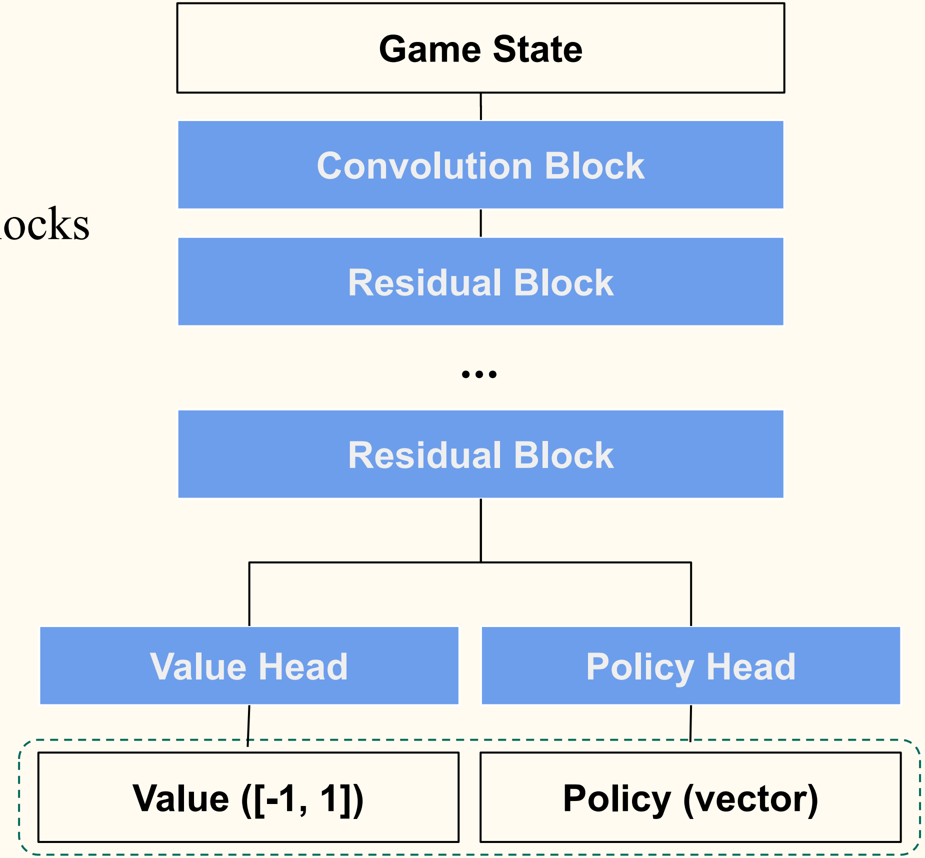
M2 MVA RL

Contents

Algorithm Review

ResNet-Based CNN

- Input: game state (8x3x3 tensor)
- 1 convolution block + several residual blocks
- Every block:
  - Convolution layers with 3x3 kernel
  - Batch normalization
  - Relu
- Value head
  - Convolution layers + fc layers
  - Outputs a scalar real value
- Policy head
  - Convolution layers + fc layers
  - Outputs a real-valued vector (8*8*4)



MCTS

- **Node**:
  - State Node: contains all information of the current game state
  - Action Node: represent the action going from its parent state node to its child state node

State Node
Parent (1 Action Node)
Children (N Action Nodes)
Game Map (np.array)
Movable Pieces (np.array)
Player (int)
GameOver (bool)

Action Node
Parent (1 State Node)
Child (1 State Node)
Action (tuple)
Piece Coordinate (np.array)
Stats (N, W, Q, P)

MCTS

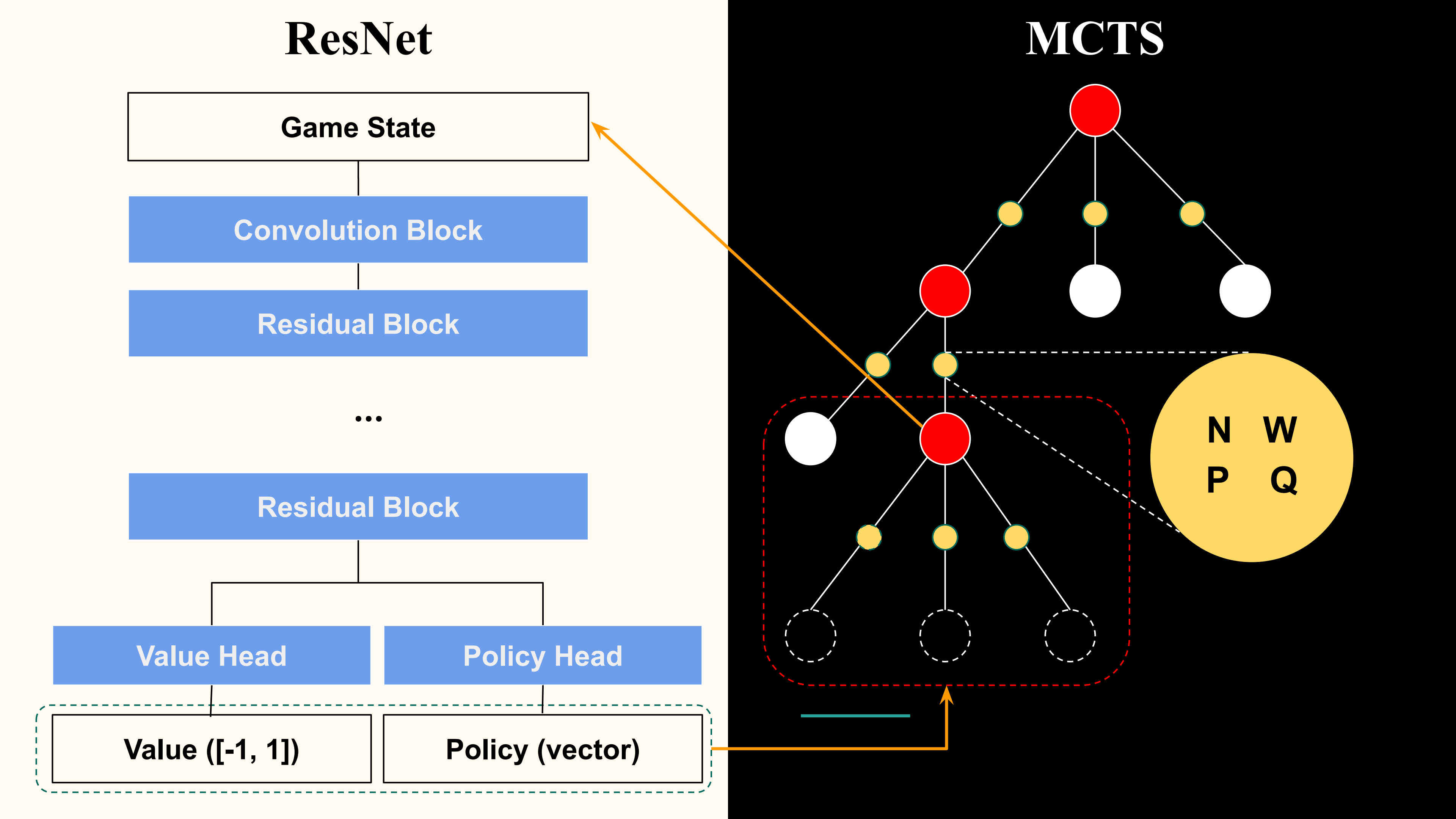
- **Basic Operations**:
  - Selection: Choose the following action
    - Deterministic Strategy: choose the node whose N is largest
    - Probabilistic Strategy: $\pi \sim N^{1/\tau}$
  - Expansion: Initialize all possible moves with
    - $N = W = Q = 0$
    - $P = \text{prior}_{\text{CNN}}$
  - Evaluation: A complete process going from the root to a GameOver State
  - Backpropagation
    - $N = N + 1$, $W = W + v$, $Q = W / N$

MCTS

- **Simulation**
  - Polynomial Upper Confidence Trees (PUCT)

$$U = Q(s,a) + c_{puct} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$$

  - Exploration - Exploitation Strategy: Choose the child whose U is largest
    - Large c_{puct} : encourage exploration
    - Small c_{puct} : encourage exploitation


ResNet
Game State
Convolution Block
Residual Block
...
Residual Block
Value Head
Policy Head
Value ([-1, 1])
Policy (vector)
MCTS
N W
P Q

Joint Training

Three components executed asynchronously in parallel

Self-play:

- Creating dataset
- Using the best neural network so far

Optimization of neural network:

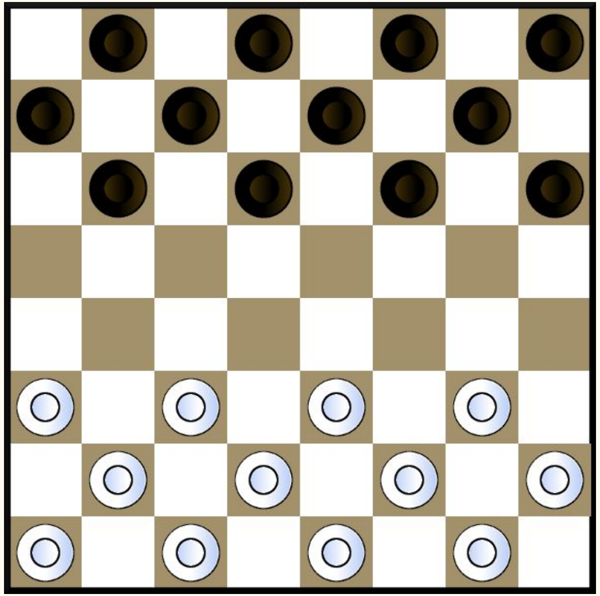
- Training neural network
- Sampling batches from recent self-play games
- Cross-entropy : policy
- Mean squared error : value

Evaluator:

- Update the best neural network so far
- Update only if the new neural network is stronger according to results of competition

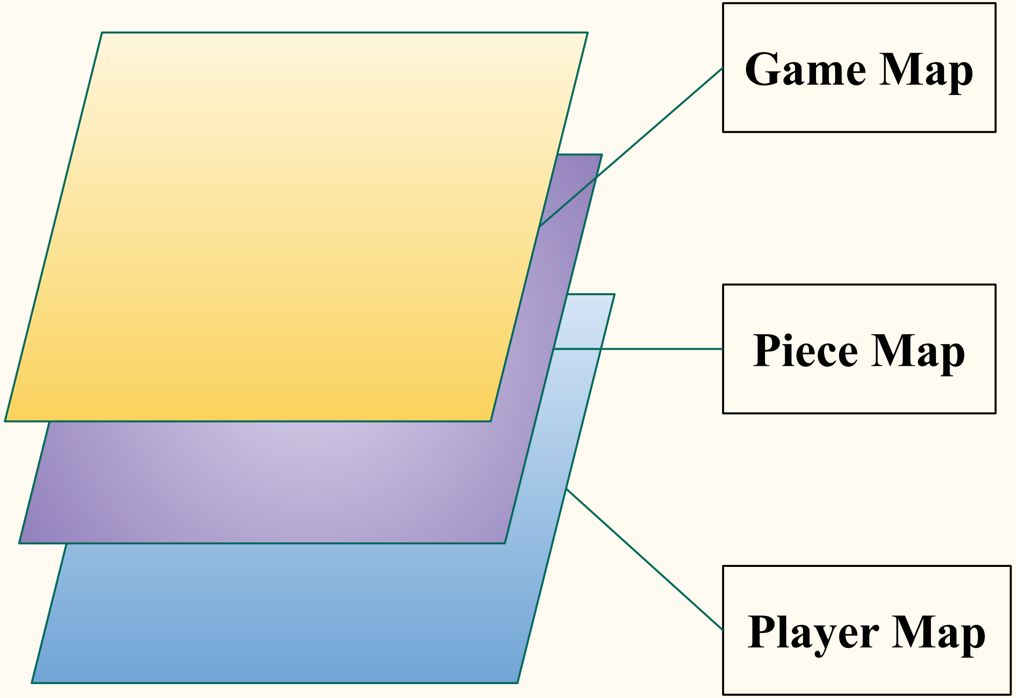
Game Design

Game Rule - English Draughts



- **Move**:
  - Single Move: Move forward diagonally to an unoccupied square
  - Jump: Eat an opponent’s piece which lies on the adjacent square. Multiple jumps are possible and mandatory
- **King**:
  - Once the piece reaches opponent’s border, it becomes a king and can move both forward and backward
- **Game Over**:
  - No piece left on the board
  - No possible moves

Game State



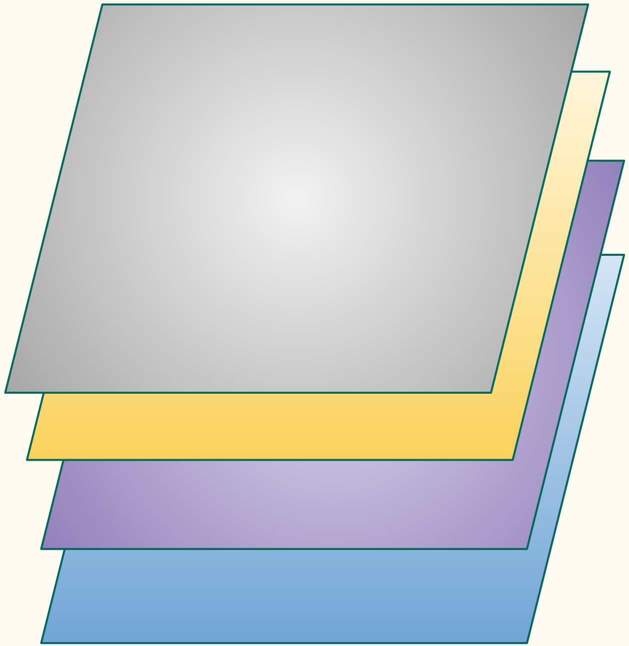
0 : No pieces
1 : White piece
-1 : Black piece

0 : No pieces / Unmovable pieces
1 : Movable pieces

All 1s if white
All -1s if black

8 x 8 x 3

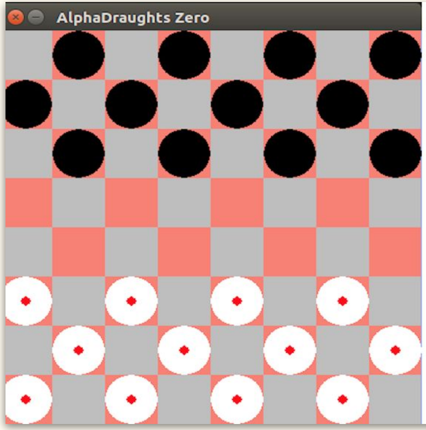
Game Policy



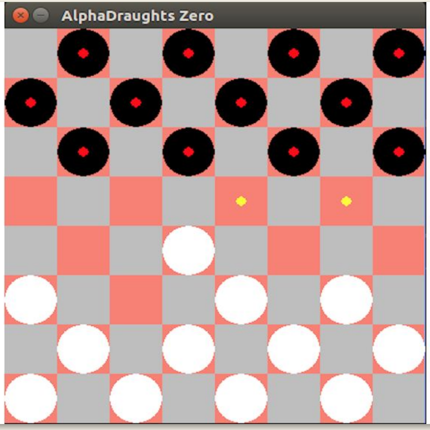
8 x 8 x 4

- The entries are all between 0 and 1, which represents the probability the piece choose the direction {NorthWest, NorthEast, Southwest, SouthEast}.
- The MCTS filters all moves and keeps only legal moves.

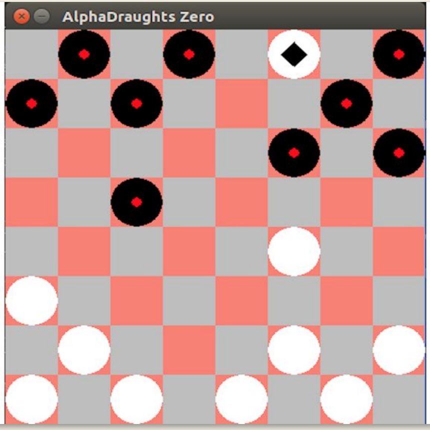
Graphical User Interface (GUI)



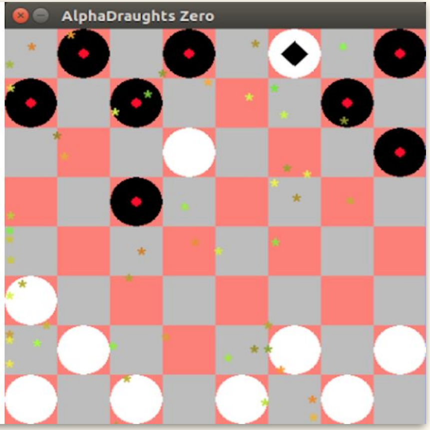
(a) Game Board



(b) Possible Moves



(c) King



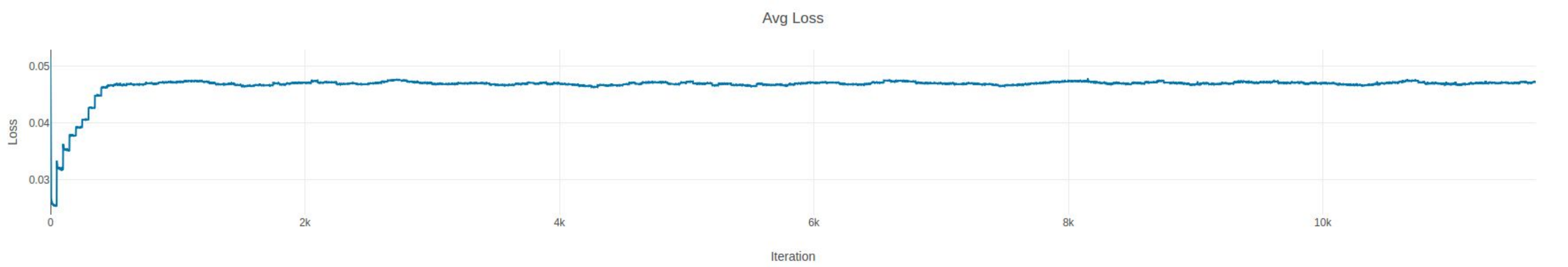
(d) Animation

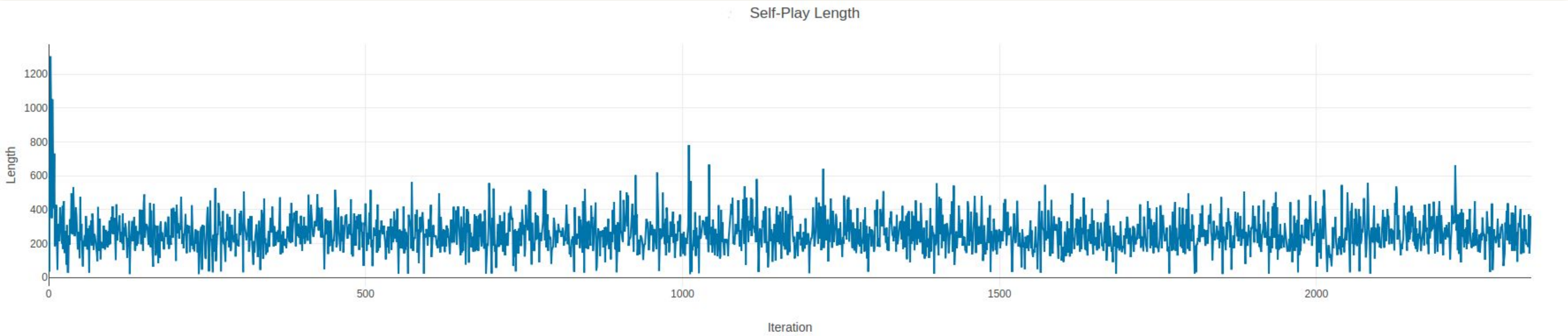
Experiment

Training

Param	Value	Param	Value
CPU	Intel E3-1231 v3	**Learning Rate**	0.01
GPU	Nvidia GTX 1070	**Self Play Time**	10
Training Time	3 days	**Number of Epochs per Iteration**	50
Simulation Per Time	20	c_{puct}	0.1

Training


Avg Loss
0.05
0.04
0.03
Loss
0
2k
4k
6k
8k
10k
Iteration


Self-Play Length
1200
1000
800
600
400
200
0
Length
0
500
1000
1500
2000
Iteration

DEMO TIME ~

Conclusion

Conclusion

- AlphaGo Zero is a efficient yet time-consuming reinforcement learning algorithm for games with complete information
- Parameter selection is not obvious
- The time of simulation is crucial to get a good model

https://github.com/Tong-ZHAO/AlphaDraughts-Zero

Thank You!