AlbaTablut

A "rising" tablut player …still rising

The approach

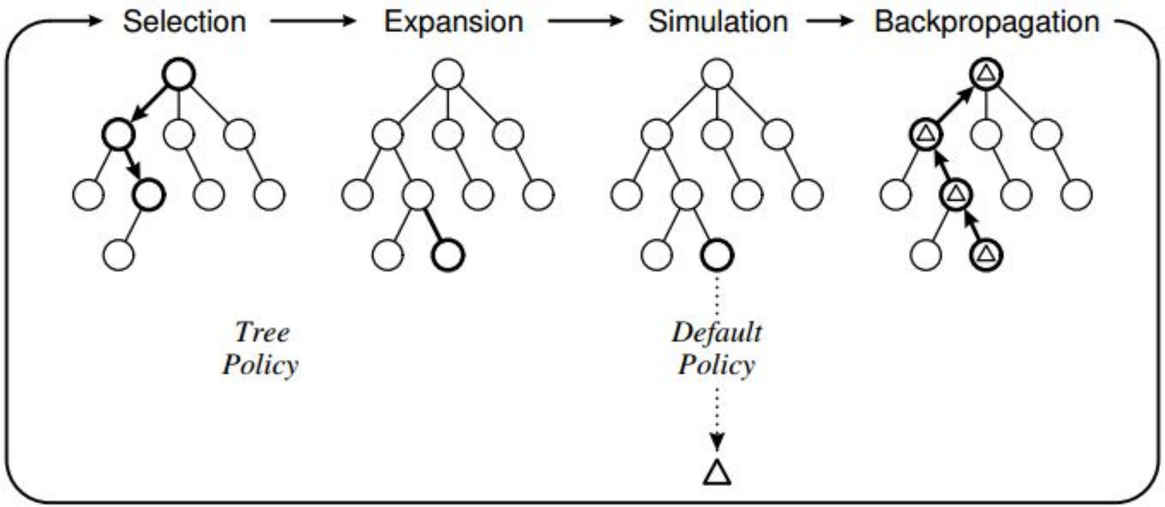
Goal: attempt to replicate the idea behind AlphaGo and AlphaGo Zero, in order to learn more about how it works!

Method:

- Monte Carlo Tree Search to explore the search space
- Neural networks to improve MCTS, trained using past years' games data

Search space exploration - Monte Carlo Tree Search

- Explore in greater depth the paths deriving from **more promising actions**
- **Evaluate how good** a newly expanded node is by simulating the game evolution
- Backpropagate the knowledge up the tree for the next exploration round

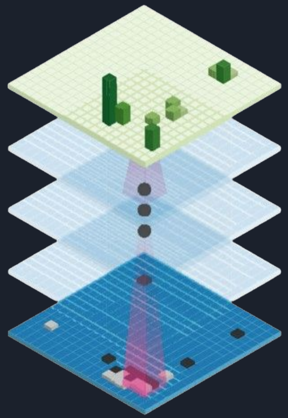


Improvements from Neural Networks

Policy network

Problem: Expansion and Selection (initially) are random, uninformed ⟶ inefficient exploration

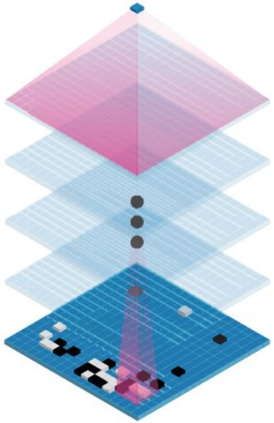
Returns a **distribution over actions**, highlighting the more promising ones, given a board state



Value network

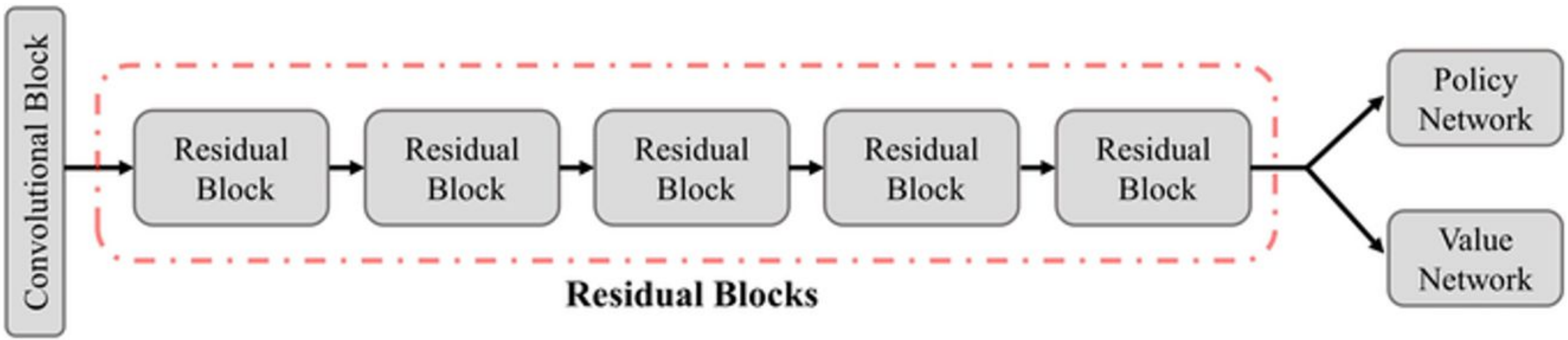
Problem: Simulation phase is costly

Predicts **win rate** given a board state, avoiding the need of simulating



Network architecture

- The two networks are actually just one network, with two different heads at the end
- 9 Residual blocks, making wide use of Convolutional and BatchNorm Layers, and of Skip Connections
- 3.1 million parameters in total



Training

1

Extract transitions from past years' games:

$$\{(board_i,\ move_i,\ win/lose_i)\}_{i=1,\ldots,N}$$

with N = 4971. If we use augmentation to exploit symmetry: N = 4971 * 8 ≈ 40K

2

Train the neural network in order to predict:

- $board \rightarrow move$ for the Policy Head
- $board \rightarrow win/lose$ for the Value Head

Actually, we are training two different networks, one for the white player and one for the black one, since the game is **asymmetric**. Both are used during the search, alternately.

Closing remarks

Limitations:

- In game performance is <u>really</u> poor, with the agent playing almost randomly
- Limited amount of data with respect to the number of parameters (and unknown quality of it) $\rightarrow$ collect more data
- Probably more time/computing power needed to be able to see learning converge to a competitive player (AlphaGo Zero trained for days on 64 GPUs...)

Possible improvements: once a decent starting player is reached, can continue training using self-play

Bottom line: what the agent has learned? Not that much. What I've learned? A lot! 😊

Thank you for the attention!