

Revisiting Influential Papers

Mastering the game of Go with deep neural networks and tree search

Authors: David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

A brief presentation by **George Fakidis**, MSc Student,
Athens University of Economics and Business

Presentation Structure



Background Information
and Related Work



Novel Contributions



Methods used



Overview of Results

Background Information, Related Work



GO is a board game, more complex and abstract compared to chess



Reinforcement Learning: Monte-Carlo simulation and tree-search, Policy and Value networks



Neural Networks: Deep Convolutional Neural Networks

Novel Contributions

Surpassing human experts in GO, a game with a significantly larger search space

Usage of CNNs in combination with Monte-Carlo Tree Search

Usage of Reinforcement Learning to improve CNN performance

Learning Pipeline

Supervised Learning based on human expert games, to predict human moves

RL value and policy networks with self-play, targeting the final outcome not prediction ability

RL network for winner prediction (in self-play)

Supervised Learning(SL) Methodology (Step 1)

Deep Convolutional Neural Network (CNN)

Aim: Search Space reduction

Target: Human moves prediction

Slightly better accuracy = substantial playing improvement

Slow to evaluate

Reinforcement Learning, Policy Networks (Step 2)

- The initial policy network is provided by the previous step
- Learning with Self-play
- Update weights to maximize expected outcome
- Wins 80% of games against the previous step network
- Wins 85% of games against the strongest open-source Go program, Pachi (without search!)
- Previous SOTA with only SL of CNNs won 11% against Pachi

Reinforcement Learning, Value Networks (Step 3)

- Goal: Estimate the value function of the strongest policy from previous step
- This NN outputs a single prediction not a distribution
- Minimise mean squared error(MSE) of the prediction and the actual outcome of the game.
- Training from complete games => Overfitting
- Solution: Generate a self-play dataset, sampling distinct positions from separate games

Selecting Actions with MCTS(Final Step)

Tree Traversal based on Maximum Action Value and a bonus that ensures exploration

$$V(s_L) = (1 - \lambda)v_{\theta}(s_L) + \lambda z_L$$

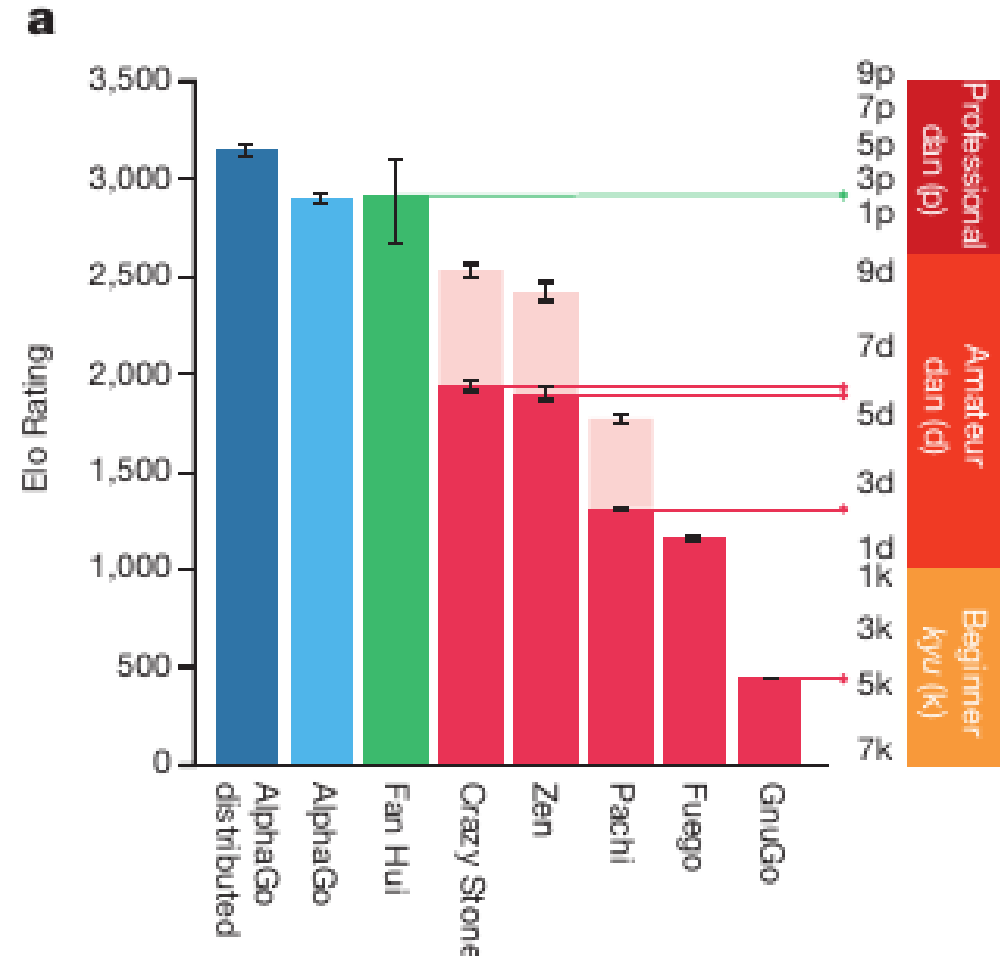
$$N(s, a) = \sum_{i=1}^n 1(s, a, i)$$

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^n 1(s, a, i) V(s_L^i)$$

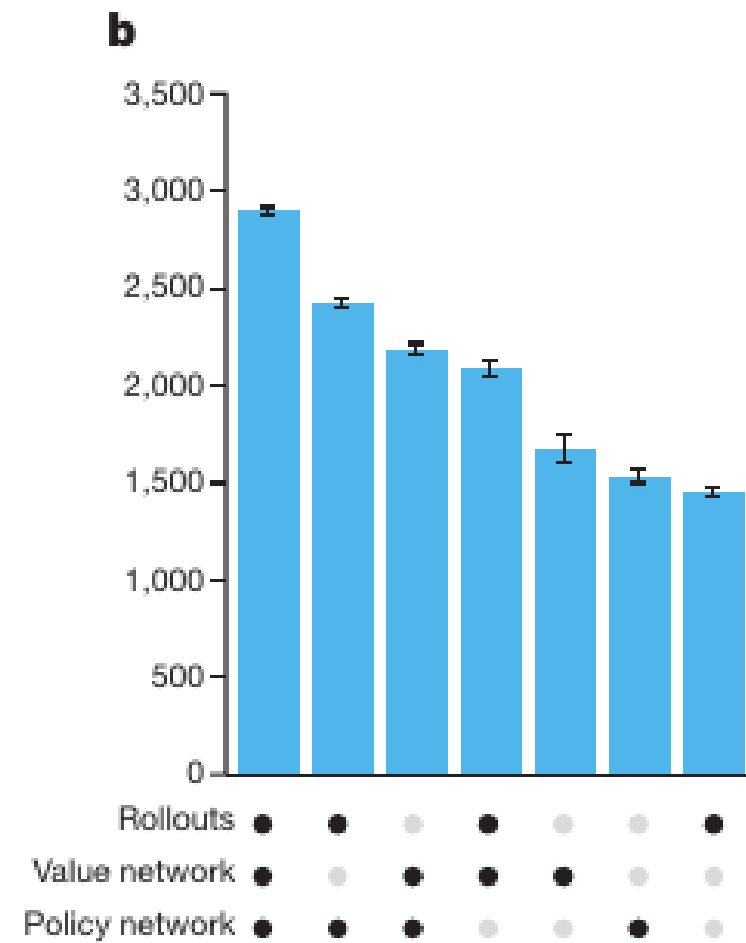
$$u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)}$$


$$a_t = \operatorname{argmax}_a (Q(s_t, a) + u(s_t, a))$$

Results



Results





Thank you !
Questions?