A review of the AlphaGo AI[1]

AlphaGo AI introduces new game-playing algorithm that achieved remarkable performance at the game of Go by combining Monte Carlo tree search (MCTS) and deep neural networks. This approach allowed drastically reducing the number of game positions it has to evaluate (and thus, computing cost) in order to play at a professional level.

As a high-level overview, this agent uses 3 policy networks and one value network with game state being depicted to them as a 19x19 pixel image:

- Policy network p_σ trained using supervised learning (SL) from games played by expert-level human players
- Fast rollout policy p_π for rapidly sampling actions
- Reinforcement learning (RL) policy p_ρ that improves p_σ by playing against other instances of AlphaGo AI
- Value network ν_θ used as an evaluation function to estimate the value of a given game state from the current player's perspective

p_σ was designed as a 13-layer neural network trained from 30M game positions. It predicted opponent moves with 55.7-57% depending on the number of input features it was allowed to use.

p_π had much poorer accuracy of 24.2% but took three orders of magnitude less time (microseconds instead of milliseconds) to generate a prediction.

p_ρ had the same structure as p_σ and its weights were initiated to the values of weights in p_σ. RL policy network was then trained by playing against randomly selected previous iteration to avoid overfitting it's play style to one specific opponent policy. RL policy trained in such a way allowed to win >80% games against an agent using SL policy p_σ

ν_θ had a similar structure to that of policy networks but it's output is a single value instead of a probability distribution. It was trained on a self-play dataset of 30M positions (each sampled from a different game to avoid overfitting) by doing regression on state-outcome pairs, with stochastic gradient descent being used to minimize mean squared error. A single evalution of game state using this function had an accuracy comparable to that of p_ρ while being four orders of magnitude faster to compute.

These 4 policy and value neural networks are used in an MCTS algorithm. This algorithm has four distinct phases: selection, expansion, evaluation, and backpropagation. It starts by selecting a move with the highest action value, then expands the leaf node and uses p_σ policy to pick a good candidate move. After that it is evaluated both by the value network ν_θ and fast rollout policy p_π. The prediction of the outcome (loss or win) is backpropagated to the top in order to update the action values of the nodes above.

These results demonstrated in this paper show that even in games with very large game space (for Go the average branching factor is ~250 and game length is ~150) modern AI can compete with and defeat the best of human players.

[1]Review is based on a paper by D. Silver et al, "Mastering the game of GO with deep neural networks and tree search", *Nature*, Vol. 529, No. 7587, (27 January 2016), pp. 484-489