Summary

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

Objective:
The game of Go has been considered one of the most challenging games to be mastered by Artificial Intelligence (AI), due to its massive search space and difficulty in evaluating positions. The Alpha Go team's main objective was to train an AI agent that could compete with the most advanced AI agents and professional human players.

Technique:
To achieve this feat, they aimed at using an approach in which they used **value networks to evaluate board positions and policy networks to select moves**.

The **policy network** consisted of **a Supervised Learning (SL) policy network, a Rollout Policy Network, a Reinforcement Learning (RL) policy network**. The purpose of the SL policy network was to learn from human players and predict what a human player would do given a board state. The SL network was trained using stochastic gradient ascent to maximize likelihood of human move selection. The Rollout policy was based on fast, incrementally computed, local pattern based features. Its objective is to guide the policy network, when searching the game tree, to focus more on good moves that a strong player is likely to search. The RL policy network has the same architecture as the SL network. It is trained by playing games against a randomly selected previous iteration of the RL policy network; and optimized towards winning the game rather than learn from expert human moves. The first iteration RL weights are initialized to be the same ones that are learned from the SL network and are updated using stochastic gradient **ascent** in the direction that **maximizes expected outcome**, which is the reward function.

The value networks are trained to focus on position evaluations using Reinforcement Learning. The neural architecture was similar to the policy networks but its outputs is a single prediction instead of a probability distribution. The network was trained using stochastic gradient **descent to minimize** the mean squared error between predicted values and outcomes. To counter overfitting the authors trained the network on a self-play dataset consisting of 30 million distinct positions, each sampled from a separate game.

The policy and value networks are **combined in a Monte Carlo Search Tree Algorithm**. In this algorithm, an edge is selected which has the max action value → The leaf node is expanded, the new node is processed once by the policy network and output probabilities are stored → The leaf nodes at the end are then evaluated using value network and by running a rollout to the end of the game with the fast rollout policy → Action values are updated to track the mean value of all evaluations in the subtree below that action. To efficiently combine MCTS with deep neural networks, AlphaGo uses asynchronous multi-threaded search that executes on CPUs and computes policy and value networks in parallel on GPUs.

Results:
The algorithm competed against two commercial (Crazy Stone and Zen) and two open source (Pachi and Fuego) programs. AlphaGo won 494 out of 495 matches (99.8%) against the agents. It was then given a 4-move handicap and still managed to get win rates of 77%, 86%, and 99% against Crazy Stone, Zen and Pachi respectively. Even with just value network (λ=0) or just rollouts (λ=1) AlphaGo outperformed all the opposing programs. With mixed evaluation (λ=0.5) AlphaGo won 95% of games against other variants.
Finally, AlphaGo competed against Fan Hui, a professional 2 dan, and winner of 2013 – 2015 European Go Championships in a formal five-game match. AlphaGo won the match 5 games to 0. This was the first time that a Go program defeated a human professional, without handicap, in a full game of Go.