Review: "Mastering the game of Go with deep neural networks and tree search" (Silver, Huang, et. al)
Mark Helms - May, 2017

Tree search techniques are a brute force method of searching for outcomes based on many possible series of moves by each player, each assumed to be acting in their own best interest. In some games, despite the fact that perfect information exists, the search space is impossibly large for today's (and presumably any time's) technology. Such is the case with Go.

Because the branches of the search tree cannot be searched in their entirety, the search range is limited either in breadth (through policies) or depth (through value functions). Previous software implemented Monte Carlo Tree Searches, which use rollout functions to pursue a single-branch, most-likely scenario, search strategy. Alpha Go uses a twist on this technique by using neural network technology to provide the value and policy for each possible move at each turn. The neural networks have learned to play by being fed data from previous expert-level play matches and from simulation play against itself and other leading Go-playing software, which implement Monte Carlo rollouts.

Much like the Monte Carlo Tree Search algorithms, the alpha Go neural networks rely heavily on simulations and historical data. Data was available from an online Go server and from random training. The advantages of the neural networks is the ability to self-select the most important set of features for each situation, without having to be explicitly programmed by a person and they're also resilient to somewhat dirty or distorted data, as they require only a "likeness" to a familiar state in order to evaluate an unknown one, often with impressive accuracy.

While training on expert data from the KGS Go Server in a supervised learning manner, the accuracy with which the policy network predicted expert moves was 55.7%, compared to 44.4% made by the best performing alternative Go software. It appears that this improvement was made in an important inflection point for performance, as it was claimed that even slight increases in accuracy resulted in big increases in playing strength.

This was further improved by the use of a reinforcement learning network, based on the policy network. This network was initialized to the final policy network and improved by training it via competition with previous iterations of the policy network. It was able to beat the supervised learning version of the network in 80% of matches and leading competitive software in 80-85% of matches, without performing searches.

To search the policy and value networks, the Monte Carlo search technique was employed. For each (state, action) pair, it kept track of the action value, the number of visits, and the prior probability (policy value). The tree was explored in a single-thread technique, exploring only one branch at each depth. The chosen branch was picked by picking the largest sum of action value and a factor proportional to the policy value and inversely proportional to one plus the number of previous visits. This encourages exploration, but rewards high action values. Because each (state, action) pair required great computation, they were queued and processed using multiple threads running on CPUs and GPUs.

In the end, Alpha Go was able to beat other Go software in 99.8% of games and finally accomplished the goal of beating Fan Hui, an expert human player, by 5 games to 0.

(Note: Not included in the paper is the defeat of Lee Sedol, a 9-Dan Go player, which occurred two months after the paper was published.)