Review of "Mastering the game of Go with deep neural networks and tree search" by Silver et. Al. in Nature, vol. 529, (2016)

This paper presents a new approach to using artificial intelligence to master the game of Go. The computer software presented, AlphaGo, is the first computer software to beat a professional Go player, which is a significant breakthrough for artificial intelligence research.

AlphaGo uses a combination of Monte Carlo tree search and deep convolutional neural networks to select its moves. A total of three policy networks and one value network were trained. The first policy network (SL policy network) was trained via supervised learning using expert human moves data. The second policy network (rollout policy network) is a smaller and faster, but less accurate policy network that is trained using the same expert human moves data. The third policy network (RL policy network) builds upon the SL policy network using policy gradient reinforcement learning. The reinforcement learning process is done by allowing the policy network to repeatedly play against previous versions of itself. Finally, the value network was trained via reinforcement learning from self-play data with 30 million points sampled from independent games. The value network provides an estimate of the value function of the best policy obtained through the RL policy network.

The search is done via a Monte Carlo tree search algorithm. The tree is traversed by selecting the action with the maximum action value plus a prior-probability dependent bonus value. When a leaf node is reached, the SL policy network is used to update the prior probabilities for each actions possible at the current leaf node. Next, the value of the leaf node is updated by a combination of the value indicated by the value network and the result of a complete rollout using the rollout policy network. The RL policy network is not used explicitly in the search algorithm, but was used to derive the value network.

The process described above is computationally very intensive. Therefore, in order to efficiently process the algorithm, AlphaGo uses both the CPU and the GPU in an asynchronous, multi-threaded search. In addition, a multi-machine, distributed version of AlphaGo was also tested to further improve computational performance.

AlphaGo was pitted against the strongest commercial and open source Go programs and achieved a 99.8% winning rate. More impressively, the distributed AlphaGo had a 100% winning rate against other Go programs and a 77% winning rate against single-machine AlphaGo. In addition, the distributed AlphaGo defeated a three times European Go champion with a score of 5-0. This success provides hope that the strategies use in this paper can be adapted and expanded on to solve other difficult artificial intelligence problems.