Summary of the AlphaGo Paper

All games of perfect information have an optimal value function which determines the outcome of the game, from every board position or state, under perfect play by all players.

The team that developed AlphaGo employ an efficient combination of policy and value networks with MCTS to reduce the effective depth and breadth of the search tree. They do this by evaluating positions using a value network, and sampling actions using a policy network.

Monte Carlo tree search (MCTS) uses Monte Carlo rollouts to estimate the value of each state in a search tree. As more simulations are executed, the search tree grows larger and the relevant values become more accurate. The policy used to select actions during search is also improved over time, by selecting children with higher values. Asymptotically, this policy converges to optimal play, and the evaluations converge to the optimal value function. However, in the game of Go, exhaustive search is infeasible.

They train the neural networks using a pipeline consisting of several stages of machine learning. First of all they train a supervised learning (SL) policy network directly from expert human moves. They also train a fast policy that can rapidly sample actions during rollouts. Next, they train a reinforcement learning (RL) policy network that improves the SL policy network by optimizing the final outcome of games of self-play. This adjusts the policy towards winning games, rather than maximizing predictive accuracy. Finally, they train a value network that predicts the winner of games played by the RL policy network against itself.

The SL policy network alternates between convolutional layers with weights, and rectifier nonlinearities. A final softmax layer outputs a probability distribution over all legal moves. The policy network is trained on randomly sampled state-action pairs, using stochastic gradient ascent to maximize the likelihood of the human move selected in a specific state.

The final stage of the training pipeline focuses on position evaluation, estimating a value function that predicts the outcome from any position of games played by using the policy for both players.

Monte Carlo tree search in AlphaGo is achieved in such a way that each simulation traverses the tree by selecting the edge with maximum action value, plus a bonus that depends on a stored prior probability for that edge. The leaf node may be expanded; the new node is processed once by the policy network and the output probabilities are stored as prior probabilities for each action. At the end of a simulation, the leaf node is evaluated in two ways: using the value network and by running a rollout to the end of the game with the fast rollout policy, then computing the winner. Action values are updated to track the mean value of all evaluations in the subtree below that action.

Once the search is complete, the algorithm chooses the most visited move from the root position.

It is worth noting that the SL policy network performed better in AlphaGo than the stronger RL policy network, presumably because humans select a diverse beam of promising moves, whereas RL optimizes for the single best move. However, the value function derived from the stronger RL policy network performed better in AlphaGo than a value function derived from the SL policy network.

Using this search algorithm AlphaGo achieved a 99.8% winning rate against other Go programs and defeated the human European Go champion by 5 games to 0.