AlphaGo: Deep neural network and Monte Carlo Tree Search for mastering the game of GO.

Go is the most challenging of the classical game for Artificial Intelligence. Before AlphaGo took place as the number one computer GO player, defeating the professional human champion of GO, the prior best implementations of that time were majorly based on Monte Carlo Tree Search (MCTS). MCTS employs Monte Carlo rollouts to evaluate the value of each state in search tree which asymptomatically over large rollouts results into providing optimal play policies and evaluation functions. To reduce the search space policies are trained to predict the expert human moves.

AlphaGo program is implemented using following stages of machine learning:

1. Supervised learning (SL) for policy network: Policy Network which gives a probability distribution over all legal moves, used to predict expert human moves, is a 13-layer supervised learning network trained from a dataset of 30 million moves. This policy network predicts the expert move at the accuracy of 57% for all inputs compared to 44.4% by the rival.

2. MCTS for fast policy network: A small set of pattern features were used to get a fast policy network which gave an accuracy of 24.2% taking only 2 microseconds compared to 3 milliseconds by SL policy network.

3. Reinforced learning (RL) for policy network: This policy network is similar in structure and initial weights as of SL policy network. The training involved playing games between the current state of policy network with a randomly selected previous iteration of the policy network. A scoring scheme evaluated score for each player and weights at every time step were updated using a stochastic gradient to maximize chances of getting expected outcome. On evaluation, RL policy network won 80% of games against SL policy network and 85% of games against its rival.

4. Reinforced learning for value function: The value function is approximated using value network which is based on RL policy network, with a change that instead of probability predictions, it outputs a single prediction. To mitigate the problem of overfitting caused by highly correlated datasets a new dataset of 30 million distinct positions sampled from the separate games was used. The resulting value function was more accurate than Monte Carlo roll out using the fast network and used 15000 times less computation.

A multi-threaded asynchronous lookahead search combines policies and value function in an MCTS algorithm which has four steps: Selection, tree transverses from root state by selecting the edge with maximum action value plus a bonus value which depends on stored prior probability for that edge; Expansion, on reaching a previously unexpanded leaf node during transversal it can be processed using SL policy network and output probabilities are stored as prior probabilities; Evaluation, at the end of simulation the

leaf node is evaluated for winner using a function which takes inputs from execution of RL value network and fast role out policy; and Backup, at end of simulation the action values and visit counts are updated to track mean value of all evaluations in the subtree below that action. At the end of above search steps algorithm chooses the most visited move from root position.
In the above implementation, SL policy network did better than RL policy network presumably because of difference in human playing on the set of diverse promising moves in contrast to RL optimizing on the single best move.

The paper presents three sets of results for two different implementations of AlphaGo (one distributed, one not): Both versions of AlphaGo significantly outperforms previously existing Go-playing AIs and are better than the best European player. Even without using all of its neural networks, just using the value network, AlphaGo performs almost as well as other AIs. Finally, by throwing more hardware at the problem, AlphaGo performs even better.