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

Goals and Procedures:
The goal of the research is to train agent to master the game of Go by using the combination of deep neural networks and tree search. Owing to the massive search space and the difficulty of evaluating board positions and moves, this research introduces neural networks and Monte Carlo Tree Search(MCTS) to computer Go that uses 'value networks' to evaluate board positions and 'policy network' to select moves.

Two 'policy networks' are used to decide moves to investigate and play. They are trained to identify promising moves from 19 by 19 image of a Go board.

- First is the Supervised Learning(SL) policy network. It consists a 13-layer deep CNN trained on 30 million positions from the KGS Go Server. It alternates convolutional layer with RELU activation layer with a final softmax layer outputs a probability distribution over all legal moves.
- The second 'policy network' is Reinforcement Learning(RL). This not only prevents the overfitting problem of the SL, but also significantly boosts the performance of the agent by making it play against itself 1.5 million times and keeping the weights of winning agents.

The 'value network' estimates the probability that current positions will lead a win or loss move for the current player and it's done so using MCTS. More specifically, from the current position at the top of the game tree, down edges to possible moves carry an action value Q that captures how good a potential move is. The game agent searches the tree for the best move in four different phases.

- Selection phase: agent chooses the edge with the highest Q and explores down that branch
- Expansion phase: when it reaches a leaf node to explore further, it creates a down branch and runs the slow SL policy network to come up with a strong candidate move.
- Evaluation phase: agent then carries out two tasks a) run the value network once to evaluate this new position, and b) use the FR network to playout from that position to the end of the game trying as many runs as possible within a time limit.
- Backup phase: after evaluation, it will propagate the information it collected during those runs and bubble it up the search tree (). If many of the runs ended badly, it will adjust the Q value for that branch down. If it often did well, it will bump it up.

After the game playing agent has used all of its allocated time evaluating many different branches, it chooses the move that yielded the highest Q value.

Results:

AlphaGo has 99.8% winning rate against other computer Go programs, demonstrating its dominance. DeepMind's research also revealed the level of computational power required to conquer such a task. The final version of AlphaGo used 40 search threads, 48 CPUs, and 8 GPUs. Though a distributed version with 40 search threads, 1,202 CPUs, and 176 GPUs was also implemented, the program's competitiveness in terms of Elo rating exhibited diminishing returns. Additionally, a human professional Go player was defeated by AlphaGo on a full-sized board for the first time in history.