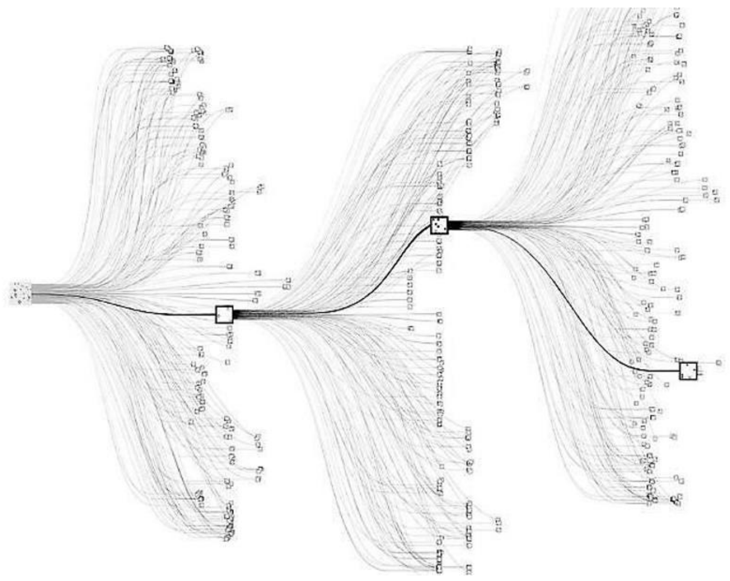
RESEARCH REVIEW

ALPHAGO BY THE DEEPMIND TEAM



All games of perfect information can be solved by recursively computing an optimal evaluation function that calculates the value of every board position in a search tree. If a game has breadth *b* and depth *d*, the approximate size of the search tree would be b^d. The search space of the classic 2 player Chinese board game called Go however, is exponentially large. The breadth of this game is approximately 250 and the depth is almost 150. That is more nodes in the search tree than there are atoms in the universe. Hence, an exhaustive search is infeasible.

The paper introduces a novel way for creating a Go playing agent by using a combination of two deep neural networks. The first 'value network' evaluates board positions and the second 'policy network' is used to select moves. The training of these neural networks happens in two stages. The first stage uses supervised learning from human expert games to learn the basic rules of the game. The second stage involves reinforcement learning, where the algorithm plays itself over and over, reinforcing good moves and penalizing bad ones, until a better version of the agent is created.

AlphaGo represents a board state as a 19X19 image. This image is fed to the two ‘deep convolutional’ neural networks to effectively reduce the depth and breadth of the search tree. Using a dataset of expert human moves provided by the KGS Go Server, the 13-layer policy network is first trained using supervised learning. The dataset consists of state-action pairs. The neural network is trained using stochastic gradient ascent to maximize the likelihood of the human move being selected. This gives quick and reasonable results for the policy network. The network has an accuracy of 57% when all input features are provided and 55.7% when just raw board positions and move history is given as input. The state of the art implementations prior to AlphaGo had an accuracy of 44%. A smaller fast policy network is trained to sample actions during a Monte Carlo rollout. This network only has an accuracy of 24.2% but runs in only a few micro seconds.

By using reinforcement learning through games of self-play, the policy network is improved. The policy network is first initialized with weights obtained from the supervised learning stage. Games are then played between the current stage of the policy network versus a random previous iteration of the policy network to prevent overfitting. In case of a win, the network is given a reward of +1 at the terminal step and -1 for a loss. The weights are then updated in the direction that maximizes a win using stochastic gradient ascent. This network wins 80% of the time against the original network that was output by the supervised learning stage. This network also wins 85% of the time against the strongest open source Go program (Pachi).

Finally, the value network is trained on state-outcome pairs using stochastic gradient descent to minimize the mean squared error (MSE) between the predicted value the corresponding outcome. Training on just the KGS data set resulted in overfitting, so the researchers decided to use a generated dataset from self-plays consisting of 30 million distinct positions. This resulted in an MSE of 0.234 on the test dataset with minimal overfitting.

AlphaGo then combines, Monte Carlo Tree Search with the policy and value networks. At a stage in the game, the policy network outputs the possible actions that the agent might take. The value network along with the fast rollout policy network in the Monte Carlo search are used to predict the value of each action. The algorithm then chooses the best possible move from the root node. The final version of AlphaGo used 40 search threads, 48 CPUs and 8 GPUs. A distributed version of AlphaGo was implemented with 1202 CPUs and 176 GPUs.

Per the final tournament against the best open source and commercial Go programs, AlphaGo won 99.8% of the games. With four handicap stones, AlphaGo won 77%, 86% and 99% of the games against Go programs called Crazy Stone, Zen and Pachi respectively. The distributed version of AlphaGo was significantly stronger. It won 77% of games against the single machine AlphaGo and 100% of games against other opponents. Finally, the distributed version of AlphaGo was pitted against a professional 2 dan European Champion Go Player, Fan Hui. AlphaGo won the match 5 games to 0. This is the first time a Go playing agent has defeated a human professional player in the full-sized game of Go.