Research Review: An analysis of AlphaGo

Introduction:

The game 'Go' is an abstract strategy board game considered to be one of the oldest games in history with the aim of surrounding more territory than your opponent. It is considered to be one of the most challenging board games in the field of AI due to the computation involved in traversing the enormous search space that the game presents to create a program with a high win ratio. AlphaGo is one of the programs built to challenge the complexity involved using modern AI techniques like deep neural networks and tree search to try and create an efficient program that combines the policy and value networks with Monte Carlo tree search (MCTS) for making this dream of AI engineers a reality. This review addresses the techniques used by the AlphaGo team along with a brief overview of the results achieved, finally concluding with an evaluation.

Techniques Involved:

Supervised learning (SL) of policy networks: The first step was to train a 13-layer (SL) policy network using 30 million positions from the KGS Go Server to try and predict expert moves in the game. This resulted in an accuracy of 57% when all input features are used, and 55.7% when using raw board positions along with move history as inputs to the network. Although not quite efficient, this network proved to be better than other research groups, which had an accuracy of 44.4% (at the time of development). It was noticed that larger networks provide better accuracy however, they were quite slow. A modified version to improve upon speed was made but its accuracy of 24.2% was far lower than what was needed.

Reinforcement learning (RL) of policy networks: The second step is to use RL to improve the policy network by playing games between the current policy network and a randomly selected previous iteration of the policy network to train it further. The outcome was – the RL policy network won more than 80% of the games against the SL policy network, and about 85% of the games against Pachi (the strongest open-source Go program) which executes 100,000 simulations per move. Without RL, the program (SL) won only 11% of the games against Pachi.

Reinforcement learning (RL) of value networks: The final step is to estimate a value function ($v^p(s)$) which predicts the outcome from position s of games played using policy p for both players. The structure of the neural network is the same as the one used in the previous steps with similar functionality, however, instead of using the expert moves, the game was played between the RL policy network and itself.

Searching with policy and value networks: To bring it together, the system combines the policy and value networks in an MCTS algorithm which selects actions to be made using lookahead search. The tree generated is traversed, starting from the root state, with each simulation selecting a state at each time step, till it reaches a leaf node. This leaf node is evaluated by the value network and then by the outcome of a random rollout played until a terminal step is reached using the fast rollout policy. At the end of the simulation, these action values along with the visit counts of all traversed edges of the tree are updated, at which point, when the search completes – it chooses the most visited move from the root position

Evaluation:

The final version of AlphaGo used 40 search threads, 48CPUs, and 8GPUs and was evaluated in an internal tournament using variants of AlphaGo and several other Go programs, with each program having a 5sec computation time per move. The results of this showed that AlphaGo won 494/495 games, while other rounds with a handicap against AlphaGo resulted in it winning 77%, 86%, and 99% against well-established Go programs like CrazyStone, Zen and Pachi. Finally, the evaluation of AlphaGo against Fan Hui (professional Go player) resulted in 5 wins and 0 losses for AlphaGo which opened up a realm of possibilities and innovative trends in AI to follow.