AlphaGo Review by Kelvin Kong
June 9, 2017

The game of Go had been viewed as the 'Grand' classic game challenge for AI, due to its enormous possibilities and difficulty of evaluating what a good move is defined as at each stage of the board. Professional Go players gained experience through years of competing against the top players around the world, and the heuristics of selecting a move is extremely complex for a computer to emulate.

The approach of computing an optimal value function by recursively searching through every possible board position is computationally infeasible as the search space for Go can reach approximately $\sim 250^{150}$ nodes. Approximate value functions that truncates the subtree such as minimax with alpha beta pruning have been successfully implemented in games like chess, checkers and Othello but it could not be applied to Go due to the complexity of the game. Monte Carlo rollouts that search to maximum depth without branching managed to provide an effective state evaluation for backgammon and Scrabble but and managed a weak amateur play in Go.

The current strongest Go AIs are based on Monte Carlo tree search (MCTS) enhanced by policies trained to predict human expert moves. The policies or value functions are limited to shallow, linear combination of input features. Leveraging the recent advancement in deep convolutional networks, AlphaGo takes the board state as a 19x19 image and use convolutional layers to construct a representation of the problem. These neural networks reduce the effective depth and breadth of the search tree; positions are evaluated using a value network while actions are sampled using a policy network. The AlphaGo neural network training comprised of several stages of machine learning. A supervised learning (SL) policy network trained directly from expert human moves, a fast policy that rapidly sample actions during rollouts. A reinforcement learning policy network is then created to improve the SL policy network by optimizing the final outcome of games by self-play. The focus is adjusted to the goal of winning games, rather than maximizing predictive accuracy. A value network is built to predict the winner of games played by the RL policy network against itself. AlphaGo is created by the combination of policy and value networks with MCTS.

The SL policy comprises of 13-layers, and was trained from 30 million positions from the KGS Go server. To improve the SL policy, RL is done by playing the current policy network with a randomly selected previous iteration of the policy network to stabilize training and avoid overfitting. The value network focuses on position evaluation, estimating a value function that predicts an outcome from a position played. The neural network outputs a single prediction instead of a probability distribution. The weights of the value network is trained by regression on state-outcome pairs using stochastic gradient descend. Predicting game outcomes from data generated by complete games lead to overfitting because successive positions are strongly correlated, but the regression target is shared for the entire game. AlphaGo combines policy and value networks in MCTS algorithm and selects an action by lookahead search. As it requires a lot more computation than conventional search heuristics, it uses asynchronous multi-threaded search that executes simulations on CPUs, while computing policy and value networks on GPUs in parallel. The final version of AlphaGo has 40 search threads, 48 CPUs and 8 GPUs. A distributed version of AlphaGo used up to 1202 CPUs and 176 GPUs, and is significantly stronger, winning up to 77% of games against single-machine AlphaGo and 100% of games against all other AI programs. AlphaGo, having an effective move selection and position evaluation and based on deep neural networks trained by supervised and reinforcement learning, has for the first time defeated a human professional player without handicap.

The challenge of playing a Go game professionally exposes the difficulties faced by AI – a challenging decision making task, a huge search space, and an optimal solution so complex that it renders direct approximation using a policy or value function impractical. This breakthrough in Go effectively

serves as a hope that it is possible for AI to reach human-level performances in other seemingly intractable AI domains.