Review of AlphaGo Research Paper

Problem Statement:

- To build a narrow AI program to play the board game ‘Go’.
- The ancient Chinese game of ‘Go’ is an unique perfect information game where the search space is larger b^d (b~ 250,d~150) than any other perfect information space game such as chess.
- As the search tree has a high branching factor, evaluation of the board positions and moves is challenging.

Summary:

Training Pipeline:

1. **Supervised learning (SL) policy network**
   - A fast rollout policy and Supervised learning policy network are trained to predict human moves in a data set of positions.
   - Alternates between convolutional layers and rectilinear nonlinearities and a final softmax layer outputs a probability distribution over all legal moves.
   - 13 layer network from 30 million positions from the KGS Go server

2. **Reinforcement learning policy network**
   - Initialized to the SL policy network and then improved to win more games against previous versions of the policy network. A new dataset is generated by playing games of self-play with the RL policy network.
   - Randomizing from a pool of opponents in this way stabilizes training by preventing over fitting to the current policy
   - For a non terminal step, the reward function returns a zero and for the terminal step at the end of the game from the perspective of the current player, the reward is +1 for winning and -1 for losing.
   - Weights are adjusted at each non terminal step by stochastic gradiant ascent in the direction that maximizes output.

3. **Value Network**
   - Trained by regression to predict the expected outcome in positions from the self play data set.
   - Outputs a single prediction as a state outcome pair (s,z)
   - Weights are adjusted by stochastic gradiant descent to minimize the mean squared error (MSE)

- To mitigate overfitting from the data set containing consisting of complete games (successive positions have strong correlation), a new self-play data set with 30 million distinct positions samples from a different game was used.

Search space reduction:

- **Depth reduced** by position evaluation using **deep convolutional network** to represent positions.
- **Breadth reduced** by using **Monte Carlo Search Trees (MCTS)** which in turn uses Monte Carlo rollouts to estimate the value of each state in the search tree.

AlphaGo combines the policy and value networks in an MCTS algorithm.

a. **Selection** Each simulation traverses the tree by selecting an edge with the maximum action value and a bonus that depends on the prior probability of that egde.
b. **Expansion** Leaf node might be expanded, new node is processed by the policy network and the output probabilities are stored as prior probabilities for each action.
c. **Evaluation** At the end of the simulation each leaf node is evaluated in two ways, using the value network and by running a rollout to the end of the game and the winner is computed with function r.
d. **Backup** Action values are updated to track all mean values of all evaluations in the sub-tree below that action

New Technique(s) Introduced:

- Combination of policy and value networks with Monte Carlo Tree search to get better performance than just using value networks or just using Monte Carlo rollouts.

Key results:

- Single machine AlphaGo is many dan stronger than any previous go programs.
- Value network provide a viable alternative to Monte carlo evaluation in Go**.** Mixed evaluation (Rollouts and Value networks) have better performance than all other variants.
- Two position-evaluation mechanisms are complementary.

Reference:

Mastering the game of Go with deep neural networks and tree search
https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf