Mastering the game of Go - AlphaGo. Research Review.

(This review is made by Ruppelt Oleksandr as a part of Udacity AI Nanodegree. You may find original paper "Mastering the game of Go with deep neural networks and tree search" by the following link - https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf)

Goals. All games of perfect information that were solved up to date used two tricks: **1)** position evaluation that truncated the search tree and replaced value function by its approximation $v(s) \approx v^*(s)$ that predicts the outcome of state s and **2)** sampling long sequences of actions for both players from a certain policy $p(a|s)$ that is a probability distribution over possible moves a from a state s (like Monte Carlo tree search). While former approach delivered superhuman performances for games of chess, checkers and othello and latter performed outstanding in backgammon and Scrabble, both were believed to fall miserably in game of Go due to various reasons. That is why aim of this research was to be able to teach a program to play Go at reasonable level of performance and, after that was achieved - to be able to learn and predict human moves and finally to beat existing programs and achieve superhuman results.

Techniques. Overview.
AlphaGo combined well-known Monte Carlo simulation with value and policy deep neural networks. Networks were designed to learn an evaluation function rather than use existing heuristics as in chess or other games. Policy networks were trained on randomly sampled state (s, a) using stochastic gradient ascent to maximize the likelihood of human move a selected in state s. It was 13-layer network trained on pictures of over 30 mln positions and predicted expert moves on the hold-out set with 57% accuracy. Improvement of policy network was achieved with reinforcement learning, where current policy network played game with it's randomly selected iteration. Final stage of pipeline was position evaluation. In practice, position evaluation of state s was estimated for a strongest policy. The networks outputs a single value, but has a similar structure as policy network. Finally, AlphaGo combines Monte Carlo search tree with policy and value networks and performs lookahead search. For each leaf node its value is estimated as a sum of neural network output and Monte Carlo rollout output : $V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$ where λ is a mixing parameter, $v_\theta(s_L)$ is network evaluation of state and z_L is outcome of random rollout played until terminal time.

Results. To evaluate AlphaGo performance it was put into a tournament against other top (by Elo rating) programs including one of the best human player Fen Hui and both top commercial and open source programs. The result achieved was beating other Go programs 494 out of 459 games. It achieved a 77% minimum win rate against opponents playing with 4 stones handicap and beating Fen Hui in 5 game tournament in the process. A mixing parameter λ that maximized winning rate was equal 0.5.