The goal of this project was to design a system, AlphaGo, that would obtain superhuman performance in the game of Go (breadth~250, depth~150 of b^d possible moves), which is much more complex than chess.

AlphaGO used a novel combination of policy network, supervised neural network(s) & reinforcement learning (RL), and value networks with Monte Carlo Tree Search (MCTS) - evaluating positions using a value network, and sampling actions using a policy network.

**Policy Networks**

To train policy network, couple of deep learning stages were involved.

The board position is passed as a 19X19 image and convolutional layers were used to construct a representation of the position. The policy network was trained on randomly sampled state-action pairs (s, a), using stochastic gradient ascent to maximize the likelihood of the human move "a" selected in state "s". In the first stage, a 13-layer policy network, called the SL policy network, was trained from 30 million positions from the KGS Go Server, which had an accuracy of 57.0% using all input features, and 55.7% using only raw board position and move history as inputs, compared to the state-of-the-art from other research groups of 44.4%.

The second stage of the training pipeline aimed at improving the policy network by policy gradient reinforcement learning (RL). The RL policy network was identical in structure to the SL policy network. When played head-to-head, the RL policy network won more than 80% of games against the SL policy network.

**Value Network**

The final stage of the training pipeline focused on position evaluation, estimating a value function that predicts the outcome from position "s" of games played by using policy "p" for both players. This neural network had a similar architecture to the policy network, but output a single prediction instead of a probability distribution. To prevent overfitting, a new self-play training data set consisting of 30 million distinct positions, each sampled from a separate game, was used. Trained value function was more accurate than Monte Carlo rollouts using rollout policy and its single evaluation had similar accuracy to Monte Carlo rollouts using Reinforcement Learning policy (but used about 15000 times less computations).

AlphaGo uses a combination of policy and value networks in Monte Carlo search tree. Game tree is searched in simulations composed from 4 phases: selection, expansion, evaluation, and backup.

**Results**

In 2015 it won against a professional human player Fan Hui, the multiple winner of European Go championships, by 5 games to 0. It was the very first time when machine defeated a professional in the full game of Go. In tournament simulation AlphaGO also won 99.8% matches against other well-known Go programs, such as Crazy Stone, Pachi, Zen and others. Distributed version of AlphaGO wins 77% of games against single-machine AlphaGO, what means that it’s not only performant but also very well scalable.