AlphaGo

The paper's goal is to create a Go game playing agent using neural networks and search trees. The team used a "value network" and "policy network" together with search tree to develop the game playing agent.

The policy network is first trained using supervised learning and then it is trained using reinforcement learning. During the reinforcement learning the network is trained against random sample of its own game play to avoid overfitting. This network outputs probability distribution over all legal moves. This initial policy network was used to play against an open-source Go program called Pachi. It had a 85% rate of success. The value network is then trained from the policy network using reinforcement learning to predict the outcome of the game.

The AlphaGo program combines the value and policy network in a Monte Carlo Search Tree (MCTS) algorithm that selects actions by lookahead search. Each simulation traverse the tree by selecting the edge with maximum action value plus prior probability for that edge. A leaf node maybe expanded. The new node is processed once by the policy network and the output probabilities are stored as prior probability for each actions. At the end of the simulation the leaf nodes are evaluated using the value network and by rollout to the end of game. The tree is updated with the appropriate value for each subtrees.

To evaluate AlphaGo the team ran an internal tournament against several other Go program (Crazy Stone, Zen, Pachi and Feugo). All the programs played against were based in MCTS. AlphaGo's winning rate was 98%. It even played in handicapped mode and won 77%,86% and 99% against Crazy Stone, Zen and Pachi respectively. The program also played against a professional Go player and it won 5 games to 0.