# Summary of AlphaGo

This paper introduces a new approach to compute Go that uses two deep neural networks to evaluate board positions and to select moves, namely 'value networks' and 'policy networks'. It also introduces a new search algorithm that combines Monte Carlo simulation with the two networks.

Training method: a novel combination of supervised learning from human expert games, and reinforcement learning from game of self-play.

1. The first stage is supervised learning of policy networks. During this stage, a 13-layer policy network, which is called the **SL policy network**, is trained from 30 million positions from the KGS Go Server. The SL policy network achieves an accuracy if 57.0% in predicting expert moves. Meanwhile, a faster but less accurate **rollout policy** is also trained using a linear softmax of small pattern features.
2. The second stage is reinforcement learning of policy networks. The SL policy network is further improved by policy gradient reinforcement learning, which is called **RL policy network**. The RL policy network won more than 80% of the games against the SL policy network. Without using search algorithm, the RL policy network won 85 of the games against Pachi, which is a sophisticated Monte Carlo search program and the strongest open-source Go program.
3. The final stage is reinforcement learning of a **value networ**k that predicts the winner of games played by the RL policy network against itself. The value network was consistently more accurate than Monte Carlo rollouts using the fast rollout policy.

Search algorithm: a Monte Carlo tree search combined with the policy and value networks. The policy network takes the board position as input and outputs a probability distribution over legal moves. The value network takes the same input but outputs a scalar value that predicts the expected outcome. The search algorithm consists of the following four steps.

1. Selection. Each simulation traverses the tree by selecting an action that maximizes action value plus a bonus that depends on a prior probability generated by the SL policy network.
2. Expansion. The leaf position is processed once by the **SL policy network** and the output probabilities are stored as prior probabilities for each action.
3. Evaluation. The leaf position is evaluated combing the **value network** derived from the **RL policy network** and the outcome of a random rollout using the fast **rollout policy**. The linear combination is controlled by a mixing parameter and AlphaGo perform best when the mixing parameter equals 0.5, which suggests that the two position-evaluation mechanism are complementary.
4. Backup. Action values are updated to track the mean value of all evaluations in the subtree.

key results that were achieved:
The win ration for Single-machine AlphaGo against any previous Go program is 99.8%. The distributed version of AlphaGo wins 77% of games against single-machine AlphaGo and 100% of games against other programs.

What's more, for the first time in history, the distributed version of AlphaGo as a computer Go program defeated a human professional player in the full game of Go, winning the match 5 games to 0.