Go, just like Chess, Checkers etc. is a game of perfect information. In idea go may be solved completely using search tree till the end of the game. If a game has big variability we just search to a certain depth and use value function to check how good is our position on this step. And if we have ideal or close to ideal value function will get an ideal game agent.

The main problem, that Go variability is so huge that determination of good or bad move from a professional player is based in some part on intuition (especially in yearly and mid game). That's why we can't build good value function, cuz we just don't have good determination of hove to qualify good position on board (again, especially in yearly and mid game).

To solve this problem Monte Carlo tree search was used. It doesn't need value function at all. It's just playing "random" game moves till end as many times as it can till it has time and set values of each move on win potential. The more simulations the more accurate it becomes. To narrow "randomness" of moves, this tree search is enhanced by policies that predict human expert moves. Using this approach, the current strongest Go programs are built (strong amateur play).

AlphaGo combines Monte Carlo tree search with policy and value networks. These networks are deep convolutional neural networks (this architecture of network also actively used in image recognition). This networks named Policy and Value networks.

Policy network used to action selection due Monte Carlo tree search, and output board map with probability distribution on each move. It trained in two step. First step - Supervised Learning on games from KGS Go Server. Second step - Reinforcement Learning where AlphaGo played games with previous iterations of itself.

Value network used for position evaluation, and output only the winning chance. To avoid overfitting new data set of uncorrelated self-play positions was created. Dataset consists from 30 million position each sampled from a unique game. Position evaluation accuracy of the value network is consistently more accurate than Monte Carlo rollouts. Also its single evaluation had similar accuracy to Monte Carlo rollouts using Reinforcement Learning policy while it used about 15000 times less computations.

Results: AlphaGo achieved a 99.8% winning rate against other Go programs. Also the distributed version of AlphaGo was significantly stronger, winning 77% of games against single-machine AlphaGo and 100% of its games against other programs.

In 2015 defeated AlphaGo win the human European Go champion by 5 games to 0, it was first time when program wins against a professional player in game of go.