Go is a two players zero-sum strategy game of perfect information where the aim is to surround more territory on the board than the opponent. Despite its simple rules, Go is considered one of the most complex games in existence. It is considered one of the great challenges of AI - to construct Go agents that can achieve expert level performance.

With an average breadth of 250 and depth of 150 legal moves, Go agents typically use heuristics of the game state to reduce the list of possible actions; thereby reducing the branching factor without having to reach terminal states. One such method that achieves this is the Monte Carlo Rollout (MCR) algorithm which limits the actions into one optimal move by taking the arg-max conditional probability of actions from simulating multiple games. The Monte Carlo Search Tree takes a step further by searching and selecting children game states to select high probability actions and thus converge to optimal play.

Ultimately the goal of a Go agent should be to learn its own heuristic function that approximates the optimal value function with minimal computational constraints. In order to improve performance over tradition Monte Carlo based methods, DeepMind developed a machine learning based program called AlphaGo to achieve expert level performance. AlphaGo reduce the depth and breadth of the game tree by using a policy and a value network to evaluate actions and winning game positions. The networks where trained on 19x19 images of the Go board using convolutional neural networks to convert the raw images into internal state/positions of the game.

Next AlphaGo was trained on a supervised learning network to accurately predict what moves an expert would execute. Given historical datasets of expert playing Go provided fast and efficient training predicting expert moves with "57.0% accuracy using all input features" and "55.7% accuracy using only raw board position and move history as inputs".

Once the supervised policy network was trained, a reinforcement policy network was initialized using the supervised network’s weights. Through additional training of the network against prior iterations of itself, the network improved its policies to take winning actions rather than accurately predict prior moves made by experts. The reinforcement network was than pitted against the original supervised network winning more than 80% of the time and 85% of games against Pachi (a open-source version of MCR).

Both the supervised and reinforcement networks learn to predict what action to take next, but need a heuristic to evaluate the quality of the game state for further use in search. Thus a value network was generated by running the RL policy network against itself to predict the likely winner.

The final version of AlphaGo combines the action generating policy network with the board evaluations of the value network in an efficient Monte Carlo Search Tree algorithm. It does this through lookahead search through action selection before storing node values and visit counts as heuristics. At each time step, the agent chooses a promising action for a given state to maximize its expect utility where leaf nodes are calculated using a combination of MCR and the value network. At the end of each time step the action values and visit counts are updated accordingly to each edge to provide more information about which actions to further pursue.

After completion, AlphaGo was better than all previous Go agents constructed so far. Out of a total of 495 games, AlphaGo beat all other Go agents except once with an impressing 99.8% winning record. When using four handicaps it won "won 77%, 86%, 99% against Crazy Stone, Zen and Pachi, respectively". Finally in a more distributed version of AlphaGo, it was able to beat the single-machine AlphaGo 77% of the time and a perfect record against all other Go agents.