Summary of AlphaGo Natural Paper

If we have complete information about every board positions than we can determine outcome of any game. For games no of possible board position can be calculated by b^d where b is games breath (no of legal moves per position) and d is its depth i.e. game length. For chess the tree contains ~35^80 and for GO its ~250^150 (the universe has only ~10^80 atoms) so calculating all move is infeasible. We can reduce this search space either by reducing depth of search or by reducing breath of the search. One way of doing so is by sampling the branching using Monte Carol Tree Search. MCTS runs simulations of games starting at the current game state and stops when game is won by one of the two players. At first, simulation are completely random but more simulations are executed, the tree grows larger and the reverent values become more accurate as each simulations. Most of strongest Go AI use MCTS and hand crafted rules designed by experts and are able achieve amateur play levels.

However AlphaGO makes an extensive use of machine learning to avoid using human experts (hand crafted rules). AlphaGo use convolutions neural network (A neural network layer is just a large matrix of numbers that takes a set of numbers as input, weighs them through a nonlinear activation function, and produces another set of numbers as output) which have achieved unprecedented performance in visual domain like image classification, face recognition. AlphaGo used two different kinds of CNN: two policy networks and one value network. Image of current game status is feed into CNN. The policy networks provides guidance regarding which action to choose, output of policy network is probability value for each legal moves (higher the values, higher the chance of winning). At first policy network was trained on 30 million positions from KGS GO server with accuracy of 57%. The policy networks were improved by letting AlphaGO play against each other and using the outcome of these games as training set, these were called reinforced-learning policy network. Second type of policy network (also called rollout network) doesn't take whole board as input, but instead only looks at a smaller window around the opponent's previous move and the new move it's considering. Removing part of the input in this way lowers its strength, but the leaner version computes about 1,000 times faster, making it suitable for reading calculations.

A value network was trained on 30 million game positions (each sampled from different game) obtained while policy networks played against itself. This network classifying potential future positions as "good" or "bad" by outputting the probability of the player to ultimately win the game. Hence, AlphaGo can decide whether or not to read more deeply along a particular variation.

AlphaGo uses a mixture of the output of the value network, two policy network along with MCTS to make more intelligent guesses about which variations to explore, and how deeply to explore them.

AlphaGO used asynchronous multi-threaded search that executes simulations on CPUs, and computes policy and value networks in parallel on GPUs as policy and value networks required several orders of magnitude more computation than traditional search heuristics.

To evaluate AlphaGo, internal tournament among variants of AlphaGo and several other Go programs was run, AlphaGO won 99.8% times. Finally, AlphaGo won 5-0 against European Go champion Fan Hui, this was the first time computer go program has defeated a human profession player, without handicap in full game of Go.

By playing against itself, AlphaGo automatically gets better and better at playing Go, which reduced tremendous effort need to handcrafted evaluation function like in deep blue. This provides hope that AlphaGo approach might adapt well to others AI problems.

Reference: Silver1 D. & Huang A. *Mastering the game of Go with deep neural networks and tree search*, in Natural, Vol 529, 484-489, (2016)