Summary of Alpha Go Paper: ***"Mastering the Game of Go with deep neural networks and tree search***"

In games with perfect information, the common way of addressing
the problem of searching the enormous game tree which is of the
order of b^d possible sequences of moves(where b is the game's breadth i.e number of legal moves per position and d is the game's depth i.e length of the game) is to define an optimal value function which can be recursively computed to determine the out come of the game.

Although this approach performs well for games with small values of d and b however, for more complicated games like Go(d=150, b = 250) searching the game tree using such an approach is not feasible.

Thus, the Authors of this paper have introduced a new approach which is based on the
application of Machine Leaning and Deep Neural Networks for searching
a complicated Game tree like Go.

The Author's main ideas is as follows:

Due to the enormous search space in the game of Go
they use two deep neural networks:
1) Value networks to evaluate board positions
2) policy networks to select moves.

These networks are trained using a 'novel' combination of
a) Supervised Learning(from Go game human experts)
b) Reinforcement Learning(from games of self-play)
c) Monte Carlo Tree Search(MTCS) for Monte Carlo rollouts which is use for networks optimization.

By Combining Monte Carlo simulations + Deep Neural Networks(Value + Policy networks),
the AlphaGo program archieved a 99.8% wining rate against other Go programs.

-- The High-Performance Game Tree Search Algorithm
The position and value search algorithm is a combination in an efficient
way, the Deep Neural Networks with Monte Carlo rollouts.
The two deep neural networks are trained using a combination of
i) Supervised Learning
ii) Reinforcement Learning

The procedure of the search of the enormous search tree(of the order of b^d (b ~= 250, d ~= 150)) problem for the game of Go is solved in the following steps:

i) Board postions is passed as a 19-by-19 image to a convolutional neural networks which produces a representation of the positions.

ii) A combination of Neural Networks is then used to reduce the effective depth(d) and breadth(b) of the tree.

The deep neural networks are applied accordingly:

a) Human experts are used to train Supervised Learning Policy Networks

b) Apply Monte Carlo Tree Search(MTCS) which performs Monte Carlo rollouts to estimate the value of each search tree. The purpose of this is to reduce the breadth of the search by sampling long sequences of actions for both players from a policy and it is used to train a fast policy network.

c) Reinforcement Learning(RL) train a policy network to improve on the Supervised Learning Ploicy Network.

d) Finally a Reinforcement Learning Value network is trained to predict the game's winner played by the Reinforcement Learning policy network against itself.

-**--Monte Carlo Tree Search(MCTS)**

Instead of the traditional minimax approach,
the breadth problem is solved by using Monte Carlo Tree Search (MCTS) method which is
use to estimate the optimal value of the interior nodes in a search tree.
The estimation is performed through 2 step approximation process.
The first approximation uses a number of Monte Carlo simulations to estimate the
value function os a simulation policy which is then use in the seconda approximation
to obtain the optimal value of the nodes.
A policy p(a|s) is defined as a probability distribution over posible moves(actions) a in position(state) s.
By sampling this probability we are sampling long sequences of actions for both players and as a result searching to a maxinum depth without branching. The probabilty distributions are produced using Monte Carlo simulations rollouts.
Averaging through several rollouts gives an effective position evaluation.
This entire procedure of using Monte Carlo rollouts is refered to as Montt Carlo Tree Search (MCTS).
Thus value of each state in a search tree is obtained using the MCTS method and with
more simulations the value become more accurate.

---Long Computation time:

To overcome the issue of long computation times which is common with Evaluatiing the
policy and value networks than for traditional recursive search heuristics,
the authors of AlphaGo use Asynchronous multi-threaded search that executes

simulations on CPUs and computes policy and value networks in parallel on GPUs.
The final version of the AlphaGo program used 40search threads, 48 CPUs and 8 GPUs.

---Performance Evaluation:

The performance of the final version of the program AlphaGo
is evaluated by creating an internal tournament among variants of the AlphaGo and several other Go programs like Crazy Stones,Zen, Pachi, Fuego. Crazy Stones and Zen are commercial wheras Pachi and Fuego and open source programs and they are based on high-performance MCTS algorithms. An additional player was
GnuGo which uses another state of the art algorithm preceeding the MCTS.

Out of 495 Go games played(each Go program allowed a computation time of 5s to make a move) a single machine AlphaGo program won 494 of the games giving it
a whooping 99.8% win rate.
The distributed version of AlphaGo program won 100% of the games against the other players and 77% against a single machine AlphaGo program.