# AlphaGo
# Deep Mind
**Sashank Santhanam**

The authors in this article introduce a new method that uses **value networks** to evaluate the board positions and **policy networks** to select the possible moves. The authors also combine this with using deep neural networks and reinforcement learning. Further, a new algorithm that **combines Monte Carlo simulation with value and policy networks** was introduced to achieve a winning rate of 99.8%.

Every game can solved by recursively computing the optimal value function from a search which contains b^d possible moves. Using two general principles can do effective searching of the game space. The first principle is reducing depth by position evaluation and truncating a search tree at a state and the other principle is controlling the breadth by sampling actions from a policy p(a|s) using probability distribution of moves a in s. Monte Carlo Search Tree(MCTS) uses monte carlo rollouts that is used to evaluate each state of the search tree and policy can be used over a period of time to select children with high values and Alpha Go uses this above method.

The first stage of the training pipeline was to predict the moves of GO using supervised learning. The supervised learning policy network alternates between convolutional layers with weights  and ReLU and the last layer is a softmax and the input to the network would be the current state of the board. The authors used a 13 layers policy network to do this phase. The network predicted expert moves on a held out test set with an accuracy of 57.0% using all input features, and 55.7% using only raw board position and move history as inputs.

The second stage of the training process was to improve the policy network by using policy gradient reinforcement learning. This networks structure was similar to the one used in the Supervised Learning network and network plays games between current policy network and randomly selected ones. The performance of this policy network was evaluated in game play. The reinforcement learning policy network dominated over the SL policy network and won more than 80% of the games and also this network beat every other competing algorithm like Pachi.

The final stage of the training pipeline involved evaluation of the positions based on a value function. The authors estimated the strongest value function using the RL policy network and this function was estimated with another neural network that has a similar architecture to the policy network, but it gives a single output instead of a probability of values. The authors also MSE and stochastic Gradient descent to do the back propagation and minimize the errors. To over come the problems associated with overfitting, the authors generated a new self-play data set consisting of 30 million distinct positions, each sampled from a separate game. Each game was played between the RL policy network and itself until the game terminated. Training on this data set led to MSEs of 0.226 and 0.234 on the training and test set respectively, indicating minimal overfitting

AlphaGo also combines the policy and values networks with the MCTS algorithm. The game tress is traversed by simulation and at each time step  an action is selected from the state and when the traversal reaches a leaf node sL at step L, the leaf node may be expanded. The leaf position sL is processed just once by the SL policy network and the output probabilities are stored as probabilities P for each legal action. Alpha Go uses asynchronous multithreaded search that executes simulations on CPU and computes policy and values networks on GPU. Evaluating the performance of AlphaGo was done by conduction an internal tournament and having several variations of it and The results of the tournament that AlphaGo ranks stronger than any previous Go program, winning 494 out of 495 games (99.8%) against other Go programs and the distributed version of Alpha Go beat the single machine Alpha go almost 77% of the time and distributed alpha GO also defeated a professional GO player.