The model will first train Supervised Learning policy network(SL) directly from expert human moves to provides fast and efficient learning updates and immediate feedback and high-quality gradients.

In the first section, it uses convolutional layers and rectifier nonlinearity to find probability distribution of all legal moves (a) from board state (s) using stochastic gradient **ascent** to maximize likelihood of human move. In SL stage, the model trained fast rollout policy to capture small pattern using linear softmax.

Next train reinforcement learning policy network (RL) that improves SL network. This helps adjust policy towards correct goal of winning the games, not maximizing predictive accuracy.

The RL network is identical to SL network. Its weights are initialized at the same value. Its opponent is selected from previous iteration of policy network. By randomizing from a pool of opponents will helps preventing overfitting to the current policy. The reward function is set at +1 for winning and -1 for losing. Then weights are update at each time step by stochastic gradient **ascent** to maximizes expected outcome.

After finished RL for policy network we then proceed to RL of value networks.

It is for position evaluation the predicts outcome from position s of games played by using policy p for both players. It has similar architecture to policy network but outputs a single prediction instead of probability distribution. The network is trained by using stochastic gradient **descent** to minimize Mean Squared error between predicted value and corresponding outcome (z). However, when using this with actual dataset, it will result in overfitting as successive position are strongly correlated, differing by just one stone, but regression target is shares across entire game. Therefore, the model generates self-play dataset of 30mn distinct position between RL policy network and itself until the game is terminated. By this way, it could mitigate overfitting problems.

The model then combines policy and value network in Monte Carlo tree search algorithm (MCTS). At each time step, action is selected from state to maximize action value plus a bonus that depends on stored prior probability for that edge but decays with repeated visits to encourage exploration. After the end of simulation, action values and visit counts of all traversed edges are updated. Once search is completed, algorithm choose the most visited move from root position.

One interesting point is the evaluation that the model utilizes both rollout network along with value network (using RL) by setting lambda = 0.5. The value network approximates the outcome by strong but slow while rollout can precisely score and evaluate outcome of game by weaker but faster rollout policy. This balance significantly improve Alpha Go while maintaining efficient use of time.

Another interesting point is that, during the game, it evaluated less position than Deep Blue did in chess match. This is compensated by selecting those positions more intelligently using policy network and evaluating them more precisely using value network.

As a result, Alpha Go can win more than 95% of the games against other variants and also beat Go European Champion; Fan Hui.