Google's AlphaGo project is a seminal work in creating a solution that outplayed the best human in the Go game. Matching human intellect has been a grand challenge for artificial intelligence and this is a step in that direction. Following is how I think they did it from my reading of the article.

There are three steps in the process. The first two steps are the evaluation phase and the third one is the decision process. Before the actual process the board is taken and passed through a set of convoluted layers to help build position on the board. Once the board is encoded and the position is established, neural networks are used to reduce the breadth and depth of the search tree. The first step is to create a policy network. This is developed using supervised learning by directly picking up moves from human games. The policy network itself comprises of 13 layers is convolutions networks with standard weights and there is a final soft max layer which provides a probability distribution across all possible legal moves. The policy network was trained using over 30 million moves.

The second step is building on top of the policy network using policy gradient reinforcement learning. The structure of this network is same as the policy network. This is done by a lot of self play. Current version of the policy network is played against randomly selected previous iterations of the same network. A reward function is used which ensure that the winning player gets a +1 and the loser a -1. Then weights are adjusted in the direction of expected outcome using a separate method called stochastic gradient ascent. When the RL network was tested against the SL network, the RL network won more than 80% of the games. The RL network was also tested on some of the open source solutions like Pachi. And the RL network won easily.

The final stage of the pipeline is the value network. This value network is similar to the policy network. But outputs a single value instead of a probability distribution. This is done using a value function that predicts the outcome of a game given the current position. Combinations of state and outcome are used to continuously optimize the weights. They deliberately stayed away from using entire games which had known results. The problem with this approach is over fitting.

Finally the policy network and the value network are used in a Monte Carlo Tree search algorithm with a lookahead search. These evaluating policies and value networks require a lot more computation capacity than traditional means.

The results of this new approach is very interesting. AlphaGo turned out to be many ranks ahead of the other open source players. The interesting part is that increasing the number of filters did not increase the overall performance of the system after a certain level. By filter 192, the optimal level was reached. The team played with multiple opponents including variations of alphaGo, Pachi, CrazyStone and Zen. The AlphaGo system won very convincingly. When distributed AlphaGo was played against a standalone version of the same, the results were similar.