Brief Summary of New Techniques in the Paper

The AlphaGo program discussed in paper "Mastering the game of Go with deep neural networks and tree search" uses a new search algorithm combining machine learning training of fast feed-forward architecture deep Convolutional Networks (CN) including Monte Carlo Tree Search (MCTS) simulations using brute-force and new techniques leveraging artificial Neural Network (NN) models trained on Value Networks (VN) and Policy Networks (PN) to win playing the Go game (alternating Markov game).

Go has a wide branching factor, a large intractable search space of possible moves at each game state, and players compete in games of around 150 moves. Go game tree node count estimated is 10^{761}.

Both VNs and PNs accept as input an image of the current whole game board state.

AlphaGo uses a self-learning Evaluation Functions (EF) for current position evaluation using a mixture of outputs from VNs (intuition) trained using regression to approximate Value Functions (VF) of the Reinforcement Learning Policy Networks (RLPN) without using terminal rewards, and PNs trained to classify moves based on pre-processed positions in feature planes using self-play simulation results (reflection), and using multi-threaded and truncated Asynchronous Policy and Value MCTS algorithm (APV-MCTS) using an implicit symmetry ensemble that randomly selects a rotation/reflection when evaluating deep neural networks.

Policy Networks (PN) and Monte Carlo Tree Search (MCTS) Simulations

PNs output a probability distribution of values for possible moves leading to a win, providing move selection guidance to reduce the tree search breadth. Supervised Learning Policy Networks (SLPN) layers are trained on millions of positions from games played by human experts for fast feedback.

RLPN weights are further optimised and stabilised by training them using the MCTS algorithm, which searches game trees by mixing in truncated Monte Carlo rollouts (temporal-difference-like learning). It simulates thousands of random self-play sequences from current game state, each to a maximum depth without branching. RLPN records statistics about likelihood of each nodes' of usage and leading to a win.

Applying an ever improving Policy of gameplay actions grows the search tree until asymptotic convergence reached for accurately select winning moves quickly for optimal play. Randomly selecting opponent iterations avoids overfitting.

The output is divided by the frequency the move was taken in current game state during the MCTS simulations by applying a trade-off that penalises often chosen actions to encourage exploration.

Value Networks (VN)

VNs evaluate board positions using convolutional layers, and output a probability the current state results in win. VNs are like self-learning EFs using regression to predict current player wins using training established by SLPN/RLPN. VNs' prediction accuracy increases near end game.

Brief Summary of Results in the Paper

Depth First Search Minimax with Alpha-Beta Pruning is not effective in Go so AlphaGo uses MCTS. AlphaGo uses Deep CNs (intuition and reflection) to outperform human experts and those relying solely on VNs (intuition) or search (reflections). AlphaGo won 80% of games using only RLPN against its SLPN. Distributed AlphaGo won 77% of games against single-machine AlphaGo.