Review of
Mastering the game of Go with deep neural networks and tree search

Varuna Bamunusinghe

Udacity

Paper in review discuss a search algorithm designed with the goal of mastering Go using deep learning neural networks. Go has a branching factor of about 250 and depth of about 150; which places the complexity of the search tree around 10^{350}. Due to the large branching factor, AI Go agents did not achieve professional ranking at the time of writing this paper; best open-source go playing agent Pachi was ranked at mare 2 amateur dan.

This paper introduced a new search algorithm which combined deep convolutional neural networks with MCTS (Monte Carlo Tree Search). New algorithm was used in AlphaGo which ultimately achieved strongest level human play in Go; a feat that was predicted to take decades. Core of the algorithm contained value networks and policy networks. V*alue networks* evaluated board positions to reduce depth of the search tree and *policy networks* reduced breadth of the game tree by sampling available actions.

AlphaGo used two step pipeline to train policy networks. First step used Supervised Learning (SL) to train the network by feeding games played by human professionals. Second step used Reinforcement Learning (RL) by feeding games played between policy networks played trained using SL. Output of the policy network is a probability distribution of available legal moves. Paper notes that using SL policy networks out performed RL policy networks, because human experts selected diverse range of promising moves while AI agents always selected optimal moves only.

AlphaGo also used a value network trained using RL. Output of the value network was a single value evaluating game state for the agent. Using SL for value networks did not work due to overfitting occurred by memorizing game outcomes instead of generalizing it. To mitigate this problem, value network was trained through RL by feeding games played between policy networks.

AlphaGo used policy network and value network in MCTS to select optimal moves through a tree search.

To achieve higher performance, AlphaGo used asynchronous multi-threads to search through game tree using CPUs and policy/value networks were evaluated in parallel using GPUs. Distributed version of AlphaGo was also introduced to scale horizontally for even more power.

RL policy networks alone won 85% of the games in a head-to-head battle with Pachi. Tournament between AlphaGo and existing go programs (which included best commercial and open-open source programs) was arranged to evaluate AlpaGo's streangth. Single-machine AlphaGo won 99.8% against other Go programs. With 4 handicapped stones AlphaGo won 77% against Crazy Stone, 86% against Zen and 99% against Pachi. Distributed AlphaGo won 77% against single-machine AlphaGo and 100% against other programs.

Then AlphaGo was put to the ultimate test of winning one of AI's grand challenges; i.e. playing go at the level of strongest human players. AlphaGo beat professional 2 dan ranked European Go champion Fan Hui in a formal 5 game match, five to zero.

During the match AlphaGo evaluated just a fraction of move positions evaluated by DeepBlue in the chess match vs Gary Kasparov. This was significant, because it meant that value network and policy network evaluated the game displaying more human like intelligence. And, even though DeepBlue required handcrafted evaluation functions, AlphaGo achieved human level play by only using general purpose supervised learning and reinforcement learning to deduce evaluations.