Nicholas Beshouri
AIND Project 2 Research Review

The authors' goal was to use the recent progress in deep neural networks to try and conquer one of artificial intelligence's grand challenges: achieving superhuman performance in Go.

Go is particularly difficult for AI both because of the size of its game tree ($b \approx 250$, $d \approx 150$) and because, unlike chess, there exist no accurate evaluation heuristics to assign values to non-terminal game states. The best Go programs attempt to solve these problems by using Monte Carlo tree search (MCTS) instead of the more conventional minimax. Rather than searching exhaustively, MCTS repeatedly plays the game to completion without branching and keeps statistics on the nodes it encounters. In each loop, the algorithm begins by descending through the previously explored tree according to a selection policy that balances exploiting known high value paths and exploring new parts of the tree in the hope of finding better ones. It continues until it finds a previously unseen leaf or is forced to expand the tree by creating one. Then it evaluates the new node using a rollout in which the rest of the game is played until completion with players either randomly choosing moves or choosing moves according to a rollout policy. The result becomes the node's value and is backpropagated to all the node's ancestors in such a way that the predicted value of any given node is its rollout value averaged together with those of its children. These values get more and more accurate as more simulations are run, converging on the values that would be assigned by a theoretical optimal value function.

Go programs using MCTS had previously achieved only strong amateur level play, even when using selection policies trained to emulate human experts. In their program, AlphaGo, the authors improved on this earlier work in two important ways. First, instead of a simple linear combination of features to guide its selection policy, AlphaGo uses a 13 layer convolutional network $P\sigma$ trained on 30 million state-action pairs from games played by human experts (a simpler, shallower version, $P\pi$, is used during rollouts). Second, the authors used state-value pairs generated by a strengthened version of $P\sigma$, Pp, to train the value network $V\theta$, which can give a fast estimate of the value for any given game state. During search, AlphaGo blends $V\theta$'s value estimate of each state's value with the estimate produced from each state's rollout, reducing the number of simulations AlphaGo needs to run in order to accurately judge the value of a move.

Testing revealed the networks to be brutally effective. Even without search, $V\theta$ was stronger than any other Go program. With search, AlphaGo beat other Go programs 99.8% of the time (or 100% of the time using a distributed version). Even more impressively, it defeated the human European Go champion Fan Hui by 5 games to 0, and did so by searching far fewer positions than Deep blue did against Garry Kasparov and without the advantage of Deep Blue's handcrafted heuristics, suggesting that AlphaGo's approach is perhaps a little more human.