pierre.hockers@student.uliege.be (s152764) o.moitroux@student.uliege.be (s152122)

INFO8006: Summary report for the reading assignment

Pierre Hockers and **Olivier Moitroux**

I. PROBLEM STATEMENT

One of the common challenges of AI is to try to beat human intelligence, to achieve better results in a specific context. A simple way to put this in practice is to confront a human player to an AI player on a given game. This has been done many times, and with increasingly difficult games, against great players. It's been a while now since we were able to create AI able to beat the best human players in a variety of games, but the game of Go has long been viewed as one of the biggest challenges for AI, at least on the field of classic games. This particular game was thought to be to complex for an AI due to it's huge search space and the difficulty to evaluate values of moves and positions.

Indeed, games such as chess and Go can be solved using recursions by finding an optimal value function determining the outcome of a game from a position if all players play perfectly. This method finds its limits when the amount of legal moves per states (breadth) and length of the game (depth) becomes too big. This is the case for chess and Go, for which exhaustive search is too expensive. The problem then becomes to limit the search with respects to the breadth and depth of the decision tree.

Depth may be reduced by using heuristics : instead of evaluating the whole tree, we stop at a state and replace the subtree by a value predicting the outcome. But in order to do that, one must be able to predict accurately the outcome of a game from a state, and Go is a very complex game, which makes the decision-making task challenging.

Acting on the breadth is also possible : all options are not good to take from a position, and using probabilities distribution amongst moves can be a way to explore only promising subtrees. The best solution until now used Monte Carlo Tree Search (MCTS) over large amounts of simulations, to get more accurate values functions. Policies of decisions (to sample actions from a state during rollouts) also improve over time.The association of these policies with MCTS allowed previous work to reach strong amateur level but the policies and value functions were too shallow and not complex enough to get any better.

Beating the European champion of Go is an interesting challenge : it gives keys to a way of abording very complex problems by an AI approach. This work can be a great source of inspiration as there are many applications like finances, constrnts satisfaction, robotics, planing optimization, ... Moreover, we have experimented several times that humans can learn new strategies by observing the top world-class AI. Observing how an AI behaves on a subject can lead to new ways of apprehending a subject or a game.

II. CONTRIBUTIONS AND RESULTS

Inspired by the excellent performance achieved by deep convolutional neural networks in some specific fields, like Atari games, the authors opted for an hybrid approach based on neural networks and Monte Carlo simulation to tackle the challenge of finding an optimal solution in a reasonable amount of time.

The Alpha GO algorithm uses a pipeline structure that allies 2 policy networks to select moves and one value network to evaluate board positions. First, a policy network p_σ is trained directly by experts human moves (a.k.a. supervised learning : SL) , then another policy p_rho is trained by reinforcement learning (RL) with p_σ as a basis (input) and by using each time older versions of itself. This policy improves the performance by adjusting the policy towards the correct goal of winning games rather than maximizing the predictive accuracy. Finally, a value network v_θ is used as a prediction tool for the winner of the game, based on simulations of the RL policy played against itself. This is a novel approach to tackle the challenge of GO. Indeed, the use of a deep neural network trained by a combination of SL and RL for move selection and position evaluation was a first. Best result arised with a combination of Monte Carlo Tree search coupled with the neural network.

- The 13 layers SL policy network (p_σ) provides fast and efficient learning updates and managed to predict experts moves with an accuracy of 55.7% when trained with 30 millions positions : a huge improvement compared to the previous best score of 44.4% of other developer teams. This leads to greater results in the end game.

- The RL policy (p_ρ) uses the same structure as p_σ and is improved by self-playing against a randomly selected previous iteration of itself in order to eliminate the problem of overfitting. The method is classic, using rewards function associating scores on victory and defeat, and using the gradient ascent in order to modify the weights to maximize the expected outcome. The results were really good, with a 85% victory rate against the best open source Go program. This is significantly better than the previous state-of-the-art convolutional network based algorithm, that only scored 11% of victory. This policy performed the best among all to predict expert moves. However this policy requires a lot of

computing power.This flaw motivated the developer to train, on the side, a less-accurate but very fast policy p_π.

- To approximate the value function, a value network (v_θ) is created, similarly to the policy network, but with only one output instead of a probability distribution. The weights of the network are then trained by regression on state-outcome pairs with stochastic gradient descent to control the mean squared error between the predicted values and the corresponding outcome. When used on the set of expert human moves, the system overfitted, memorizing situation rather than generalizing them. This is why a new set was created. This custom set was build upon positions that had no relation to each other and came from different games. This reduced greatly overfitting.

These methods are then combined all-together in a Monte Carlo Tree Search. As the tree is traversed by simulations, to each state visited, an action a is sampled based on 2 criterias, one being the action value, the other one being a bonus to encourages the algorithm to explore more states. Indeed, this bonus increases with the prior probability of other simulations, but decreases with the number of simulations that explored this node earlier. When an unexplored node, called leaf, is reached, it can be expanded evaluated. The evaluation is carried out by a combination of the value network and by the outcome of a random rollout using the weaker policy p_π for faster computation. At the end of the simulation, the algorithm chooses the most visited move from the root position.

This way, AlphaGo achieved a 99.8% victory rate on 495 games in a competiton against other Go programms and defeated the european champion by 5 games to 0, an unpreceded achievement. With a four handicap stone, it was also able to win between 77% and 99% of its matches, depending on the opponent program.

This implementation, however, has a cost in computation time and the use of a asynchronous distributed system with approprAIte number of GPUs and CPUs was required.

III. DISCUSSION

The work presented in the paper is a real advance in the field. Without any lookahead search, the neural networks can play go at the level of previously best Monte Carlo Tree Search algorithms. The combination of Monte Carlo Tree search simulation with value and policy networks proved to be a more than viable solution to tackle the problem. A quite ingenious way of forcing the algorithm to explore other solutions without allowing it to explore too much and impact the performance also greatly impacted the results presented above.

But, while being an incredible advance in the history of AI, Alpha Go still suffers some problems and limitations. For example, it is not able to beat the world champion of the game. One of the first problems that comes to mind is that Alpha Go requires very high computational power and lots of hardware to train and predict the next move under reasonable time constraints. Even if the distributed version of the algorithm performs very well and is able to beat Fan Hui, the paper highlights a lack of efficiency, and reducing the power consumption and training time would be an appreciable feat. Alpha Go also requires a huge dataset of over 30 millions experts'moves to train its SL policy network. This requirement can be skipped as it has been proven by the new implementation of Alpha Go Zero that won 100 to 0 games against the world champion, and which doesn't require any dataset whatsoever[1].

[1] Demis Hassabis, David Silver. *AlphaGo Zero: Learning from scratch*. Deepmind blog, October 2017.