INFO8006: Summary report for the reading assignment

Maxence Gilson[1] and **Julien Bolland**[2]
[1] *maxence.gilson@student.uliege.be (s162425)*
[2] *julien.bolland@student.uliege.be (s161622)*

I. PROBLEM STATEMENT

The game of Go was the hardest game to implement in artificial intelligence for computer scientists. However a solution has been found by mixing the Monte Carlo search tree algorithm with the value and network policies. Those points are going to be explained further in this text.

As a first step, we believe it is important to understand why it was so difficult to win when playing Go using artificial intelligence. Indeed, Go is a game in which players play each in turn and this type of game can usually be solved by computing over and over the best value of a function in a search tree. The complexity of this type of algorithm is b^d, where b is the amount of legal moves and d is the game length. However, this is a big issue for the game of go as b = 250 and d = 150. Therefore it seems almost impossible in term of computing time.

Nevertheless, two different solutions have been found to diminish this problem.

1. It could be possible to decrease the depth by evaluating each position in order to predicts the outcome of the different sub-trees. And therefore, we would replace each sub-tree by an evaluation of the state's outcome. Even if it produced extremely good players in games such as chess, it wasn't effective for Go.

2. We can reduce b by searching which move has the biggest probability to be made by an expert at a state s. It is called the prior probability and helps the algorithm to play expert moves. This lead to a weak amateur player of Go level.

The Monte Carlo Tree Search is a tree search which uses Monte Carlo roll-outs in order to approximate the value a state is worth in a search tree. Those roll-outs are found using two functions, the policy and value functions. The policy function to give us the probability that a move would be done by an expert player of GO. Concerning the value function, it is able to evaluate the board position and to predict the outcome at a certain state. Sadly, all of this wasn't enough to resolve the artificial intelligence Go problem.

A new discovery, the deep convolutional neural networks is at the cutting edge of the artificial intelligence. This works with neurons intertwining in order to form a representation of an image. We use this neural network to increase the value and policy functions efficiency. For the game of go, we train this neural network in a specific way.

Firstly, we run it through a supervised learning of policy network p_θ and through a fast policy p_π, both at the same time. Secondly, we train it through a reinforcement learning of policy network p_ρ. Then, we train it thanks to a reinforcement learning of value policy v_θ. We can conclude that AlphaGo is a combination of the MCTS (Monte Carlo tree search), the policy network and the value network.

II. CONTRIBUTIONS AND RESULTS

The AlphaGo algorithm is based on both the MCTS algorithm and on the value and policy neural networks. In order to have an efficient neural networks, Deep Mind had set up different stages to train it.

A. Supervised learning of policy network

They first trained a supervised learning policy through human expert moves. The policy network has been trained on pairs of state-actions to predict the expert moves. This could lead us to narrow down a small list of sensible moves. Therefore, the AlphaGo algorithm would be able to predict the probability that a move would have be done by a human. A this state of the training, it was already beating the former state-of-art algorithm concerning Go. However DeepMind decided to implement another version of this algorithm as it is too slow to work correctly with the MCTS algorithm. Thus, they had to create a new version which had to balance between the computing time and the accuracy of the given probability. In order to create this supervised learning, they used a stochastic gradient ascent. This gradient ascent means that the goal is to make a maximization of the computed probability. And we, consequently, have a policy network which fast but still very precise.

B. Reinforcement learning of policy network

Later, they trained a reinforcement learning of policy networks. The procedure is easy to understand. They simply take the SL (supervised learning) policy and let it train against itself over and over. However, by doing that, it is able to beat only one single opponent. Consequently, each time the neural network is improved, this slightly different version is saved. By saving the different versions of the neural network, we'll dispose of a beam of slightly different playing styles. So our program can now play randomly against the different previous saved versions.

Furthermore, we need to know that its goal is only *to win* or *to lose*. It doesn't matter for the program if it wins by 1 or more points. Finally, the RL (reinforcement learning) improved network beats the SL network

as well as other good Go playing programs. So it proves that doing this reinforcement learning of policy network improves tremendously the algorithm's efficiency.

C. Reinforcement learning of value network

This training, which is the last one, is used to ameliorate the evaluation of the outcomes at a state s. This is possible thanks to the value network. This value function has been created to tell us how likely we are going to win depending on a chosen board position. If we want this to work ideally, we need a perfect player. However, we don't have such a player, so we use our best player as of now, which is a player following the RL policy network.

Nevertheless, trying to predict all the different outcomes would lead to overfitting. Therefore, DeepMind found a way to shallow down the number of outcomes. Indeed, a large of outcomes were similar as, from one move to another, it only differs from one state. Hence, to avoid overfitting, only the states which are very different from each other will be taken into account. It resolves the overfitting problem. This value function is much more efficient than the Monte Carlo roll-outs and almost as good as the Monte Carlo roll-outs using the RL policy. However, it does 15000 less computations than the RL policy.

D. Searching with policy and value networks

For the moment, we have trained both policy and value functions. Now, we are going to include them into the MCTS algorithm since the MCTS algorithm chooses the move to do, depending on the further different outcomes of a tree search. However, the different outcomes are too plentiful, so we combine this algorithm with both functions named above to create AlphaGo.

Practically, the functions are involved in the MCTS algorithm to narrow down the possible moves we would like to roll-out at each step. Indeed, even if we saw earlier that the value function is more efficient, we'll still use both of them.

The policy function will give us the probability for all the moves to be done by experts and the value function will tell us how likely we are to win when doing a particular move. The mixing parameter is when we use both functions to choose the best move. Moreover, after every single simulation in the search treee, you update your database (which is the search tree in this case) in order for our simulations to improve themselves. At the end, we'll choose the best move.

Something which is interesting is that the SL policy network has better performances than the RL policy network with AlphaGo. However, using the reinforcement learning of the value network is still more efficient than using the supervised learning.

E. Evaluating the playing strength of AlphaGo

To have a good idea of AlphaGo's performances, we set it against other commercial and open source programs. Some of those programs are also based on the developed MCTS algorithm. As predicted, AlphaGo beat up all the other programs by winning 494 games on 495. Even when playing games with four handicap stones, AlphaGo won the majority of the games.

Afterwards, they tried evaluating positions using only the value network or only the monte Carlo roll-outs. Even when using only the roll-outs, AlphaGo beats most of the other Go programs. However, the AlphaGo program is at its best when using both the value network and using the roll-outs. This means that both are complementary.

The final test was to set a game of Go against a professional Go player. DeepMind chose Fan Hui, a professional 2 *dan* as well as a triple European champion of Go between 2013 and 2015. Therefore, Fan Hui and AlphaGo competed in a five-game match. As predicted, the AlphaGo program won 5-0. This is an important moment in artificial intelligence as it is the first time in history that a professional Go player was defeated by a program.

III. DISCUSSION

DeepMind has created a program based on a combination of deep neural networks and tree search. AlphaGo, the name of this program, has achieved one of the artificial intelligence's greatest challenges of the moment. Its major problem was the search space. Indeed, adversely to the Deep Blue program, which is working thanks to a handcrafted evaluation function, AlphaGo is based on neural networks trained through the value and policy functions which rely on supervised and reinforcement learning. Therefore, by associating the Monte Carlo roll-outs with the neural networks it is able to narrow down more efficiently the different possible moves and evaluate them more precisely. This is much closer to human moves than the Deep Blue program.

Finally, all of those improvements lead to a professional level in Go, which was unthinkable for a artificial intelligence before a decade. We can now believe that other domains can be developed thanks to this new technology.

Advantages and shortcomings of the paper

Thanks to this article, the basis of the AlphaGo program is understandable. However, we believe that if we didn't any basic knowledge in artificial intelligence, it would had been much more difficult to understand the different neural networks and their training.

Furthermore, this paper says that the AlphaGo program is a major breakthrough in term of artificial intelligence. However, it doesn't explain what could be simplified in our everyday life with this kind of breakthrough. They could at least give us an example of how it could be used in the future.

Finally, we thought that some parts could have been given with more explanations. For example, in the *Supervised learning of policy networks* part, it is said that they used a 13-layer policy network. Why have they used a 13-layer network and not more, or less? This number had been given without any further clarification.