Towards Super-Human Artificial Intelligence in Go by Further Improvements of AlphaGo

Tapani Raiko
Aalto University, The Curious AI Company
Helsinki, Finland

February 1, 2016

Abstract

Artificial intelligence for the game of Go is approaching the level of top human players due to the announcement of the new program AlphaGo by Google DeepMind. This paper lists five development ideas to further improve it. The ideas are note tested in practice, since AlphaGo is not available as open source, and reimplementing and retraining it would require a significant amount of programming effort and computational resources.

Introduction

AlphaGo [9] has for the first time won a professional Go player in an even match. There is still some improvement to be made before the top human playing level is reached, and even if it was reached, the AI could still be improved. In the following, we describe five improvement ideas for AlphaGo. To keep the text short, we assume that the reader has knowledge of the Nature paper [9] and the concepts of deep learning and Monte Carlo tree search.

1 Multi-Task Learning

All the information in a trained neural network is based on the information in the outputs of the training data. Once the network is able to produce the outputs in the training data, the learning stops.

Considering the value network with just one bit of information as output per game is very poor in information, so it required a huge amount of training data (30 milion games) to train. One could help the situation by training the RL policy network and value network together as one multi-task network.

Several other auxiliary tasks could be included to make the training gradient even richer. You could predict the expected score (in points), and the variance of the score.

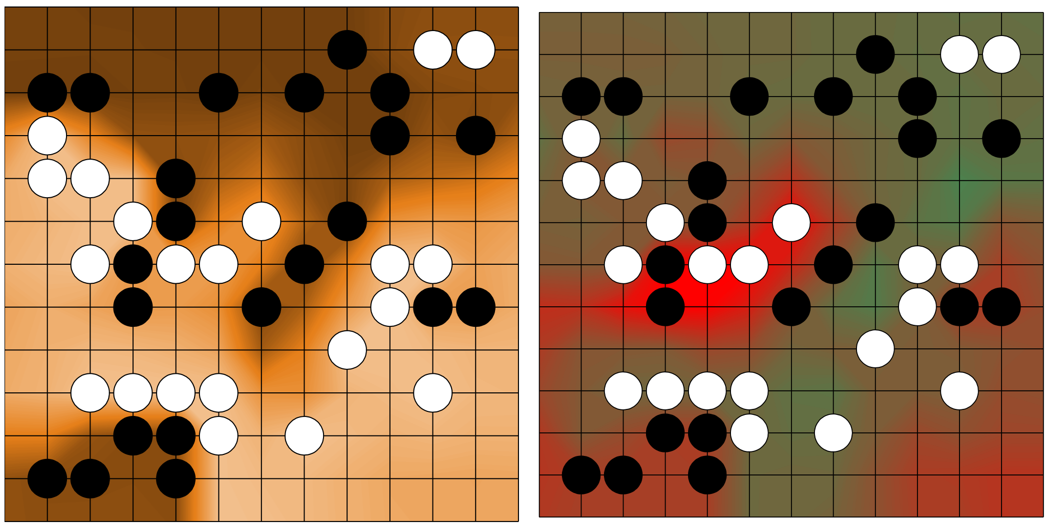

Figure 1: Possible richer auxiliary outputs for training the value network. Left: Who controls each point of the board at the end of the game (averaged over rollouts). Right: The covariance between controlling a point and winning the game (over rollouts). This indicates which parts of the board are important (marked as red). Image reprinted from [6].

You could predict who controls each point of the board at the end of the game. Assuming several rollouts from the same position, you could predict the covariance between controlling a point and winning a game. Figure 1 shows visualizations of the previous two ideas. The list could go on: Predict whether a point stays empty (becomes an eye), whether two points will be connected in the end etc. An output with the size of the board has 361 bits of information, which is much richer than the one bit. Some of the auxiliary output could perhaps be connected already to a mid-level layer.

Auxiliary outputs have been shown to be very useful just to perform better on the primary task [8], but below there are ideas of how to use them in the search tree.

2 Online Knowledge

The game in progress could have a difficult corner fight that the value network has trouble with, while look-ahead or rollouts will solve it. The errors made by the value network are repeated systematically all over the search tree, when moves are made in other areas than the corner. There could be a way of generalizing what is learned while growing the search tree, following ideas of [2]. One could include some online parameters to the value network (for instance additional local bias terms) that would be adapted from scratch during the growth of the search tree. Nodes can train their parents, and rollouts can train the corresponding value functions. Using the auxiliary outputs from

Section 1 would speed up this adaptation. This of course comes with a cost of having to run back-propagation during search, however, if the online parameters are only limited to the last layers, one would not have to run the back-propagation all the way through.

3 More Patterns for Same Computational Cost

The first layer of the network could have much more features without increasing the computational time. A unit on the first hidden layer would have a chance of activating only if there is an exact match of a local 3×3 pattern, that is, there would be a separate set of first-layer hidden units for each 3×3 pattern. There are $3^8 = 6561$ patterns where the middle point is empty, and one would only need to do computations for the features of the matching pattern. It is well known that 3×3 patterns already contain a lot of information about the move [4, 1].

4 Iterative Inference

Evaluating the strength of a group on a Go board involves evaluating its neighbouring groups (and neighbours' neighbours). It is clear that such an inference requires several steps. Raiko [5] runs iterative inference over a Markov network formed by explicitly describing the relations between the groups (See Figure 2). Silver et al. [9] use a deep network with 13 layers as the inference steps. Could it be that it is important to have many processing steps rather than many layers with their own parameters?

NADE-k [7] is a model that might have 21 layers that actually repeat the same 3 layers 7 times, that is, it is running an iterative algorithm for 7 iterations. A similar idea could be applied to AlphaGo. One could share parameters of layers 3&4 with layers 5&6 and 7&8 to run an iterative procedure for three iterations. If this looks promising, one could further extend to other iterative structures, e.g. following multi-prediction deep Boltzmann machines [3].

5 Better Prior

It is noted [9] that the RL policy network is stronger than the SL policy network, but still SL policy network performs better as the prior bonus in search. If best play is not the best goal for training the prior bonus, could you come up with the proper goal? Here is one heuristic option: Use a combination of some of those auxiliary outputs in Section 1. For instance, a move that increases the variance of the score is something aggressive that disrupts the flow of the game (such as a cut or an invasion). Such a move might not always be a good idea for winning a game, but those moves are definitely the kind to keep an eye on while searching.

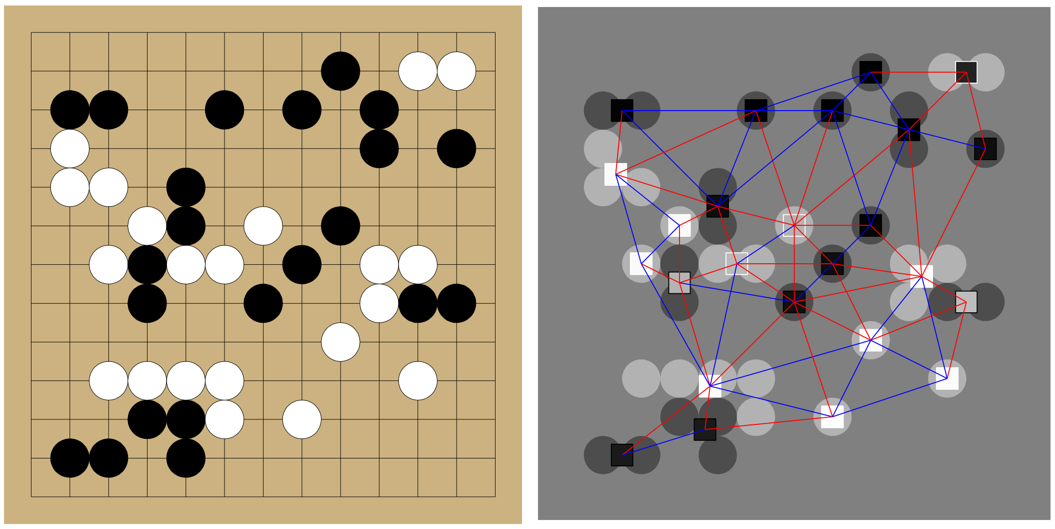

Figure 2: Left: Go game in progress. Right: Groups with their expected owner as the colour of the square (dead groups have the opponent colour, weak groups are grey). Pairs of related groups are connected with a blue line if the blocks have same colours and with a red line when the blocks have opposing colours. The lines also represent the structure of the implied Markov network. Image reprinted from [6].

References

[1] Bruno Bouzy and Bernard Helmstetter. Monte-Carlo Go developments. In *Advances in computer games*, pages 159–174. Springer, 2004.

[2] Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.

[3] Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Multi-prediction deep Boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 548–556, 2013.

[4] Tapani Raiko. The go-playing program called Go81. In *Proceedings of the Finnish Artificial Intelligence Conference, STeP 2004*, pages 197–206, 2004.

[5] Tapani Raiko. Nonlinear relational Markov networks with an application to the game of Go. In *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 989–996. Springer, 2005.

[6] Tapani Raiko. *Bayesian inference in nonlinear and relational latent variable models.* PhD thesis, 2006.

[7] Tapani Raiko, Yao Li, Kyunghyun Cho, and Yoshua Bengio. Iterative neural autoregressive distribution estimator (NADE-k). In *Advances in Neural Information Processing Systems*, pages 325–333, 2014.

[8] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3532–3540, 2015.

[9] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, (529):484–489, 2016.