TDT4265 Final project

Vsevolod Karpov 748010
Nicolas B. Carbone 768648

March 23, 2019

Project proposal

The goal of the project is to use reinforcement learning to train a CNN to play a downscaled 9x9 version of the board game Go. See also: https://en.wikipedia.org/wiki/Go_(game) The rules used for deciding the winner of the game will be Chinese rules, also known as Area rules. See here

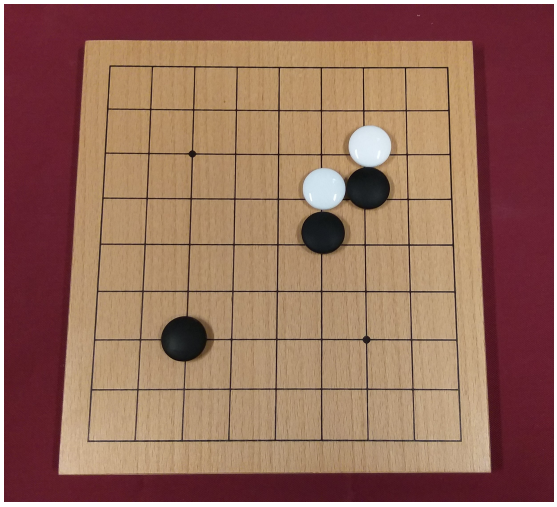

Figure 1: A standard 9x9 board of GO

Relevant literature

The motivation behind the project stems from Google Deepmind's AlphaGo, the first computer program to beat a 9-Dan professional (highest possible rank) Go player without handicaps on a full-sized 19x19 board [1]. The paper behind AlphaGo will be relevant for the project and can be found here: https:

[1] https://www.bbc.com/news/technology-35785875

//www.nature.com/articles/nature16961. Their paper on the improved version, AlphaGo Zero, will probably be more relevant as it is does not include any supervised learning, but only reinforcement learning starting *tabula rasa*. The paper on AlphaGo Zero may be found here. Further, their references contain a lot of useful information and are likely to be read as well.

Method

The method for the project will be as follows.

1. A CNN is created and duplicated 50 times (the number may be changed in the future).
2. Each CNN gets a random adjustment to its weights.
3. Each CNN plays against every other CNN as white and black.
   (a) The CNN will produce a 9x9 output where each value is between 0 and 1. The highest value that corresponds to a legal move is chosen as the next move. For instance, if highest value is at 4,3 and 4,3 is legal, 4,3 is played. If 4,3 is not legal but the next biggest value is at 5,2 and 5,2 is legal, 5,2 is played e.t.c. The network must also be able to pass if it finds no move that will increase its position.
4. Each CNN is evaluated based on a fitness function.
5. 50% worst (25) CNNs are deleted.
6. 50% best (25) CNNs are duplicated.
7. Repeat from step 2.

Primarily we were thinking to use pure reinforcement learning. Therefore no data sets will be used. Based on that we want to see if it is at all possible to get a CNN that plays decently, taking into account that we will not be looking forward or doing any form of search tree evaluation. If this seems to not work, a form of three search will be considered. Papers found on the subject used some form of Monte Carlo Three Search (MCTS), see for instance the paper on AlphaGo Zero. Note that they also implement a version called "Raw network" which performs fairly well without a MCTS.

Further experimentation with different CNN topologies will be performed to see what produces the best result. Finally we want to see how adjusting the fitness function from a function that evaluates win/loss ratio to a function that also takes in account the point difference at the end of the game. Will a system trained to win by a bigger margin perform better than a system that only cares about winning?

For evaluation we were thinking of letting the resulting CNN play against a "classic AI" that you can find online. We will also play against it ourselves but we are not very good so it probably will not reveal how good the CNN is. It should be mentioned that the amount of experimentation with CNN topology and fitness function depends on how long the training takes.

Current status

As of now, we have found a functionally python library that has most of the Go game logic and we have implemented some functions necessary for integrating a CNN on top of it. The possibility to pass a turn and the final score when a game is over was not implemented. This is under work as of today. The python library can be found here

Remaining work

The work so far has focused on a simple implementation of the game. After the game has been developed, the network according to the method description will be implemented. Different topologies will be implemented in order to see what performes better in the training process.