DEPARTMENT OF
Computer Science
TCU COLLEGE OF SCIENCE & ENGINEERING

AI2Go

Authors: Kien Nguyen, Quang Truong, Kimon Vogt, Megan Phan
Advisors: Dr. Liran Ma, Dr. Ze-Li Dou

SciCom Let's Talk Science

What Is Go?

- Go is a Chinese board game for two players, in which the aim is to surround more territory than the opponent. One player plays with white stones, and the other plays with black stones. The official grid comprises 19x19 lines, containing 361 points.

Strategy

- Strategy deals with global influence, interaction between distant stones, keeping the whole board in mind during local fights, and other issues that involve the overall game. It is therefore possible to allow a tactical loss when it confers a strategic advantage.

- Novices often start by randomly placing stones on the board, as if it were a game of chance. An understanding of how stones connect for greater power develops, and then a few basic common opening sequences may be understood.

Area Scoring

- A player's (final) score is the number of stones that the player has on the board plus the number of empty intersections surrounded by that player's stones.

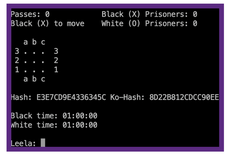
Background

- *October 2015.* **AlphaGo** by **DeepMind** defeated the European champion Fan Hui, becoming the first program to achieve superhuman performance in Go.

- *March 2016.* **AlphaGo** defeated Lee Sedol, the winner of 18 international titles, 4-1 in a five-game match, shocking the world.

- **AlphaGo** was then improved to a newer version called **AlphaZero**, a stronger AI program that self-trained, with NO prior knowledge, after being told only the rules of the game. From then, the strength of AI kept climbing at an astonishing rate.

- **Gian-Carlo Pascutto**, a computer programmer who works at the Mozilla Corporation, combined the **Monte Carlo Tree Search** and **a neural network** into building the world's most successful open-source Go engines - first **Leela**, then **LeelaZero** - which mirrored the advances made by **DeepMind**.



- The tree search and the neural network, through **Reinforcement Learning**, improve one another during training to produce better move decision and stronger self-play in the next iteration (self-play game).

Goals

Based on the open-source engine, we plan to take an alternative path utilizing LeelaZero: finding the optimal results/solutions on different board sizes from 3x3 up to 9x9. Because of symmetry, there is a difference between an even and an odd n x n board size. Therefore, we aim to:

- Obtain **optimal solutions** for board sizes from 3x3 to 9x9.
- **Integrate** the project idea into teaching in mathematic topics.

Challenges

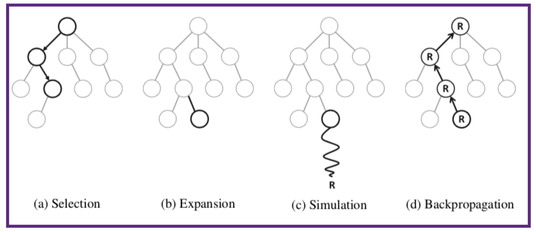
The project is a very challenging one because of the Reinforcement Learning learning curve like the concept of Monte Carlo Tree Search. Some challenges we may face are:

- The complete **understanding** of the source code.
- The **investigation** and **modification** of the source code so that it works as desired.
- The **reformatting** of the weight files so that we can train the model on different board sizes.

Monte Carlo Tree Search (MCTS)

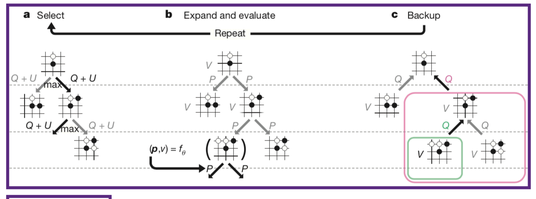
- **MCTS** is a go-to algorithm for writing bots to play **discrete** (individually separate and distinct moves and positions), **deterministic** (every move has a set outcome) **games** (players competing against one another) with **perfect information** (both players see everything). **MCTS** selectively tries moves based on how good the game-in-progress is in an **initial state**, and it is the bot's turn to play.

Visualizing MCTS



MCTS In AlphaZero

- Each edge *(s, a)* in the search tree stores a prior probability *P(s, a)*, a visit count *N(s, a)*, and an action value *Q(s, a)*.
- Each simulation starts from the root state and iteratively selects moves that maximize an upper confidence bound *Q(s, a) + U(s, a)*.

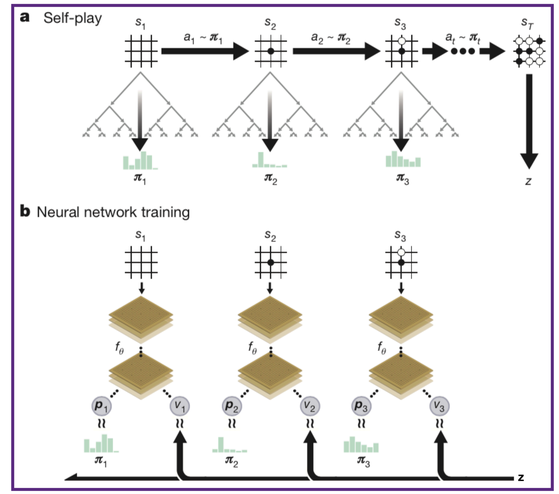


a. Traverses the tree until reaching a leaf node.
b. The leaf node expands to simulate next possible moves (looking ahead).
c. Action value *Q* is updated to track the mean of all evaluations *V* in the subtree below that action.
d. Once the search is completed, the search probabilities **π** are returned.

The Single Neural Network

- The neural network in AlphaZero takes the **raw board representation *s* of the position** and its history as an input, and outputs both move probabilities ***p*** and a value *v*, $(\boldsymbol{p}, v) = f(s)$.
  - The vector ***p*** represents the probability of selecting each move a (including pass), Pr(a I*s*)
  - Value *v* is a scalar, estimating the probability of the current player winning from position *s*.
- This network combines the roles of both policy network and value network into a single architecture.

Self-Play Training Pipeline



- The *main idea* of the reinforcement learning algorithm is to use these search operators repeatedly in a policy iteration procedure: the network's parameters are updated to make the **move probabilities** and **value** (***p***, v) more closely match the improved **search probabilities** and **self-play winner** (**π**, z).

Conclusions and Future Work

- We successfully converted the original 19x19 board size into smaller odd ones. We are currently using the original weight file provided by Pascutto to run the program and trying to **investigate** on the structure of the weight files and on how to **modify** them to fit the training with different board sizes.
- We will continue the investigation on the weight files and the modification on the source code to achieve the goal. When that is done, we will move on to **train** the program on different board sizes, **record** the results to find out the optimal solutions, and **compare** with the mathematically proven results.

Technologies Used

C++ is the programming language used to write the LeelaZero program.

Python is the programming language used to train the program using TensorFlow

References

Go Game Overview: https://en.wikipedia.org/wiki/Go_(game)
LeelaZero GitHub Repository: https://github.com/gcp/leela-zero
AlphaZero Overview: https://deepmind.com/documents/119/agz_unformatted_nature.pdf
The New Yorker's Article About AlphaZero and LeelaZero: https://www.newyorker.com/science/elements/how-the-artificial-intelligence-program-alphazero-mastered-its-games
Monte Carlo Tree Search Overview: http://mcts.ai/
Reinforcement Learning Guide: https://skymind.ai/wiki/deep-reinforcement-learning

Acknowledgements

The AI 2 Go team would like to thank the following professors:
- **Dr. Liran Ma** for advising the team throughout the project and holding interesting meetings for us to discuss challenging problems and to learn new concepts about Artificial Intelligence.
- **Dr. Ze-Li Dou** for inspiring the team through his idea proposal of discovering the optimal solutions and of integrating the Go game into teaching.
- **Dr. Bingyang Wei** for showing the team the way to approach the project in the most direct and efficient way.

We also thank the CSE and the Computer Science Department for providing us the GPU support.