Go Engine

1st Anthony McEntire
dept. of Computer Science
Middle Tennessee State Univeristy
Murfeesboro, Tennessee
utahmcentire@gmail.com

1st Brooks Herrin
dept. of Computer Science
Middle Tennessee State Univeristy
Murfeesboro, Tennessee
blh6t@mtmail.mtsu.edu

1st Eric Ramsay
dept. of Computer Science
Middle Tennessee State Univeristy
Murfeesboro, Tennessee
ebr2x@mtmail.mtsu.edu

1st George Maddux
dept. of Computer Science
Middle Tennessee State Univeristy
Murfeesboro, Tennessee
gdm2z@mtmail.mtsu.edu

1st Matthew Gabriel
dept. of Computer Science
Middle Tennessee State Univeristy
Murfeesboro, Tennessee
gabrielmatthew20@gmail.com

***Abstract*—Go is an ancient Chinese game that has been played for thousands of years. This game is hard to implement into a neural-net as there are many possibilities for play.**

This is a study into using a neural-net to play go at a beginner level. A convolution network was used to take in data and provide weights for a turn by turn analysis of where to play a piece.

The results of this study was a 23 percent accuracy on the data set of 8600 games from the app Go Quest. However, the study has resulted in being unplayable in the current iteration.

This project proves how important convolution networks are to the study of neural networks. And how hard it is for smaller computers to parse through the training of a neural net.

***Index Terms*—introduction, background, results, discution**

I. Introduction

"Gentlemen should not waste their time on trivial games – they should study go." Confucius, c. 500 BCE

Go is an ancient game, its history spanning back over 2500 years to its first written mention in China. It was a game played by all strata of society, peasants and nobles alike, a trait no doubt owing to the extreme simplicity of its ruleset. [1].

The rules are as follows: there are two sides, black and white, which play on a board consisting of a number of intersections, traditionally 19 by 19, but sometimes smaller. Black makes the first move, placing a stone on any empty intersection which they desire. From then on it alternates play between black and white. When a stone is played, it can have up to four "liberties", the empty squares adjacent to the stone, as shown in figure 1. When all of a stone's liberties are occupied by an opponent's stones, the stone is captured, as is shown in figure 2. If a stone is adjacent to another stone of its own color, the stones are connected and form a group. Stones in a group can not be taken unless all of the liberties of that group are filled, as is shown in figure 3.

These rules, concerning placing and capturing stones, encompass the majority of the game. Additionally, there are two rules concerning illegal moves. Suicide, that is placing a stone at an intersection where it would both have no liberties and capture no stones, is prohibited. Additionally is the rule

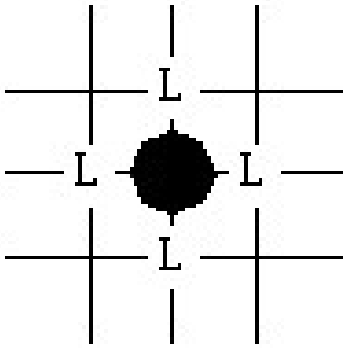

Fig. 1. The l's in this picture show where the liberties are. [2].

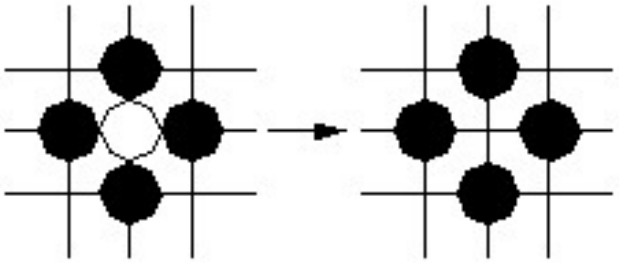

Fig. 2. This figure shows how a piece is taken. [2].

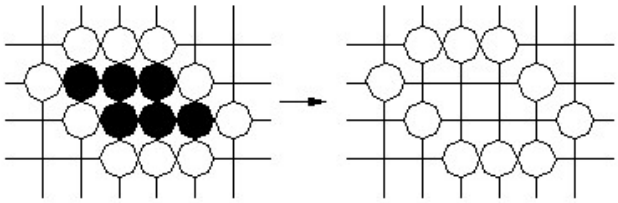

Fig. 3. If the pieces are in a line all liberties must be taken. [2].

of ko which states that a move cannot be made which would cause a repeat of a past position. The final rule of play is that players are allowed to pass their move if they wish. If both players pass, the game proceeds to scoring.

There are slightly different rulesets used for scoring, some award points for having captured an opponent's stones, some do not, but both rulesets share the principle of area or territory. Players receive a point for each intersection they "control", that is intersections adjacent to or surrounded by their stones. It can sometimes be unclear when scoring the game to determine what groups of stones may be alive or dead. This territory is shown in figure 4.

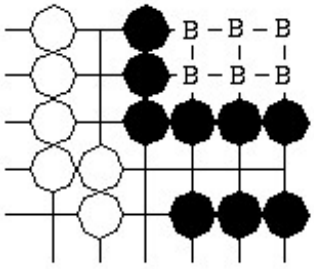

Fig. 4. The b's here show where points would be counted. [2].

The complexity which arises from these simple rules, as well as the absence of a very clear scoring method has long been a stumbling block for Go playing programs, whether they were traditionally programmed or utilized a neural net. For many years it was assumed that a program could never beat the best Go players. This assumption was proved false in 2016 with the defeat of Lee Sedol by DeepMind's AlphaGo, a neural network which was the inspiration for this endeavor. [3].

It is the goal of this project to create a neural net capable of performing at a level comparable to amateur human players on a 9x9 board.

II. Background

There are many neural nets designed to play video games and board games alike. These are all studies on if a computer can or will take over menial tasks or more complex things. Some of these nets include OpenAI which beat Dota 2 professionals. [4]. This was done by having it play itself for 10,000 years over 10 months. [4]. It has a win rate of 99.4Another more closely related Ai program is Alphago [3]. This is a program that "combines an advanced search tree with neural networks" and reinforcement learning(cite 2). AlphaGo is spoken about more in the book by Max Pumperla and Kevin Ferguson. This book goes through how reinforcement learning is used to provide better weights in the neural net as well as structural parts of the neural net that were used in this project. AlphaGo was able to beat the top players in the Chinese arena with ease. Since then, different versions of AlphaGo have been released with different data sets to start off at. Leela is also a program that plays go and is able to be downloaded. It also has the ability to output the most probable moves in any given situation. With the complexity of this game and the fact that others have been able figure out similarly complex games, we wanted to see what could be done with less complex machinery and a smaller team.

III. Methods

The data set that was used in this project was an 8600 game set played by high level users on the app Go Quest. These games were broken into over 400,000 individual positions, which were then ultimately encoded as a one-dimensional array consisting of the 81 intersections on the board, as well as an input for whose turn it was in the position.

Initially, the one-dimensional array was taken and put into sequential basic architecture that had 8 dense hidden layers each with a thousand weights in each. This, however, proved to be ineffectual and only produced a slightly better than random chance of 4 percent.

This then led to an implementation of a more complex way of inputting the data with a two-dimensional array that was taken and shaped into a 9 by 9 board. This was coded in a way so that later if needed the board could be expanded for training a different variant. This was done to cut down on the input that was being received to speed up the training process. This gave a small increase in accuracy and speed.

The two-dimensional array then allowed for the use of a convolution network and treating the array as an image. At this stage to speed up training, the data sample was down sampled and maxpooling was implemented to reduce the size of the hidden layers. As well as drop out layers were added to reduce the possibility of generalization. Lastly the input was turned back into a 1D array. This was to help with feeding data from the drivers into the array. The only thing that was really changed was a layer inserted to transform the array into a 2d array.

IV. Results

As of the deadline of this paper, a competent model has not been produced.

In the first model, there was an accuracy of 23 percent. As show in the model 5.

In this model it shows that over the course of training the model consistency learned and was able to win against the 8600 games twenty three percent of the time. However, this does not count exterior games. These are shown by the above graph in the fact that the training accuracy reached above .2, or 20 percent, at one hundred epoch range. Epoch being how many times the net is trained. The other thing to notice in the graph is the model loss. This shows that there was no modular collapse.

The newer model has not been trained as much but it is being trained as of right now. The model has a percentage of over twenty percent at the eighth epoch. As well as a gradual decent on the loss graph showing that the new model has a lot of potential. This should lead to a far better chance of

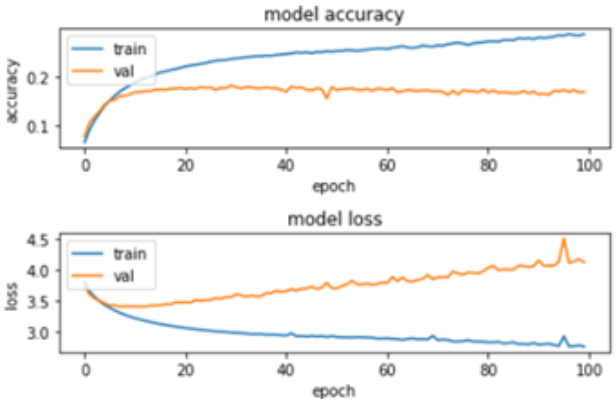

Fig. 5. This is the older model with the higher acuracy.

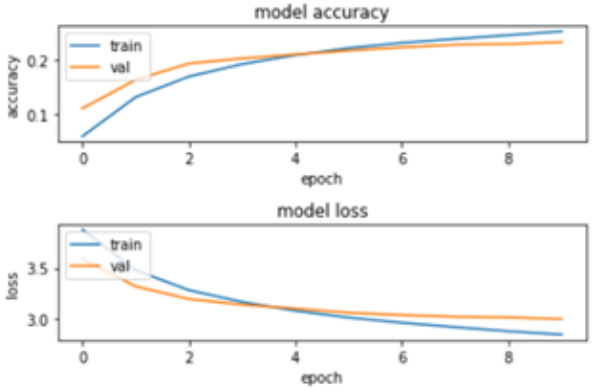

Fig. 6. this model is what is being worked on now.

winning against the data. The belief is that this model could reach the point that it could consistently beat a human. [5].

V. Discussion

As there is a board already built to play Go, the project is close to being finished. From this project the use of convolution net work is very important to the study of board games. As it goes from a four percent accuracy to a twenty three percent accuracy in less time. As the project continues, the different ways that have already been used will be combined until there is a workable solution.

The challenge that this Game presents to a neural net is hard to compute with the resources a student has available to him. As such, the hardest part of this project has been the resources that were used to train the network such as Jupiter lab and Google Colab.

[4].

References

[1] R. Eno, "The analects of confucius," 2015.
[2] J. Burmeister and J. Wiles, "Cs-tr-339 computer go tech report," 2013.
[3] "Alphago: The story so far," 2016.
[4] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1912.06680
[5] M. Pumperla and K. Ferguson, "Deep learning and the game of go," 2018.